SocialWeaver: Collaborative Inference of Human Conversation Networks Using Smartphones

Chengwen Luo
School of Computing
National University of Singapore
chluo@comp.nus.edu.sg

Mun Choon Chan
School of Computing
National University of Singapore
chanmc@comp.nus.edu.sg

ABSTRACT

Understanding how people communicate with one another plays a very important role in many disciplines including social psychology, economics, marketing, and management science. This paper proposes and evaluates SocialWeaver, a sensing service running on smartphones that performs conversation clustering and builds conversation networks automatically. SocialWeaver uses a hybrid speaker classification scheme that exploits an adaptive histogram-based classifier to non-obtrusively bootstrap the *in situ* speaker model learning. The conversation clustering algorithm proposed is able to detect fine-grain conversation groups even if speakers are close together. Finally, to address energy constrain, a POMDP-based energy control scheme is incorporated.

We evaluate the performance of each component in SocialWeaver using more than 100 hours of conversation data collected from conversation groups with sizes ranging from 2 to 13. Evaluation shows that accuracy of 71% to 92% can be achieved for various conversation modes and up to 50% of the energy consumption in SocialWeaver can be reduced through the POMDP-based scheme. Evaluations of SocialWeaver in both controlled and uncontrolled settings show promising results in realistic settings and potential to enable many future applications.

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]: Distributed applications

General Terms

Design, Human Factors, Experimentation

Keywords

Mobile phone sensing, Conversation clustering, Conversation network

1. INTRODUCTION

Understanding how people communicate with each other plays a very important role in many different research disciplines, including social psychology, economics, marketing and management science. For example, researchers have shown that the communication pattern in an organization greatly affects its efficiency, innovation and employee well beings [27]. There have been lots of research on extracting human communication patterns and these research efforts largely focus on communications through phone calls, emails [3], or online social networks [13]. On the other hand, face-to-face human conversations, which capture tremendous amount of information on human interaction patterns, have received less attention.

In this paper, our objective is to design a system that can detect conversation groups and infer human conversation networks. Such information can be constructed through different approaches. One approach is through the use of user survey, which suffers from high monetary cost and poor accuracy due to biased human factors [30]. Another approach is to use sensing technologies to perform automatic inference. Sociometer [5] and Multi-Sensor Board [30] are two such platforms. In these platforms, special devices are worn by participants to capture their conversation patterns. While these platforms provide insights on the design of spontaneous conversation detection systems, they only perform raw conversation detection and provide no fine-grain conversation clustering. In addition, the need for special-purpose hardware devices, centralized offline training, and high computation cost limit the usability and widespread deployment of these systems.

In the design of our conversation sensing system, we aim to meet the following objectives. First, the system should be able to detect simultaneous conversation groups even if these conservation groups are physically close to one another. Second, the system should be robust and can work accurately in different environment settings. Third, the system should not need specialized hardware and require minimum user intervention. Finally, the system must respect user privacy.

Our Contributions. We have designed and implemented *SocialWeaver* , a smartphone-based conversation sensing system which can perform conversation clustering and construct conversation networks among the users. It also allows easy deployment and maintenance as the application can be deployed in the form of *downloadable app.* SocialWeaver meets the design objectives in the following way:

(1) By collaboratively sharing information within the prox-

imity group, SocialWeaver exploits conversation clustering to detect fine-grain conversational interactions and differentiate different conversation groups even if the speakers are close together.

(2) SocialWeaver requires no pre-training. An adaptive histogram-based classifier and collaborative learning are used to bootstrap the *in situ* speaker model learning. As a result, the system is non-obtrusive and can dynamically adapt to different environments and speakers.

(3) SocialWeaver uses a Partially Observable Markov Decision Process (POMDP) based energy control scheme to reduce energy consumption by up to 50%.

(4) SocialWeaver respects user-privacy. Each smartphone only identifies audio samples from its owner. No audio samples or acoustic features are stored. Instead, only the timestamps of the owner's speaking durations are used.

We have implemented SocialWeaver on Android platform and run it on both low-end and high-end smartphone models which have different hardware capabilities. Evaluation of SocialWeaver shows that it performs well in realistic environment settings and is able to perform conversation clustering and construct conversation networks with high accuracy.

The structure of the paper is as follows. We present related work in Section 2 and in Section 3 the overall design and operations of SocialWeaver. We then describe the proximity module in Section 4, speaker classification module in Section 5, collaboration module in Section 6 and clustering module in Section 7. Section 8 presents our POMDP-based energy control scheme. Section 9 presents the evaluation results and Section 10 the discussions. We finally discuss limitation and future work in Section 11 and conclude in Section 12.

2. RELATED WORK

People-centric Sensing. Recently, we have witnessed an increasing popularity in people-centric sensing [4] based on smartphone platforms. Applications of people-centric sensing cover many different areas including healthcare [14], transportation [11], environment monitoring [9], and social networking [20]. To the best of our knowledge, SocialWeaver is the first opportunistic conversation sensing system running on the smartphone platform that performs both speaker classification and conversation clustering.

Speaker Identification. Existing speaker identification methods mostly apply supervised learning methods which require training for each speaker [25, 26]. Such algorithms require the total number of speakers to be static and each speaker must provide labeled samples, therefore making the system less practical and deployable. To perform speaker identification on resource-constrained smartphones, SpeakerSense [17] builds a prototype using a heterogeneous multiprocessor hardware architecture to support energy efficient continuous background speaker identification. To address the challenge of training data acquisition, speaker models can be learned from daily phone calls, one-to-one conversations or shared from other users. This approach reduces the training data collection effort, but the performance degrades if the training data are collected from environments that are different from the current setting. In contrast, SocialWeaver assumes no *a priori* information about any speaker and trains *in situ* speaker models. SocioPhone [15] exploits a pure volume-topography-based approach to detect speakers. This method is light-weighted and accurate

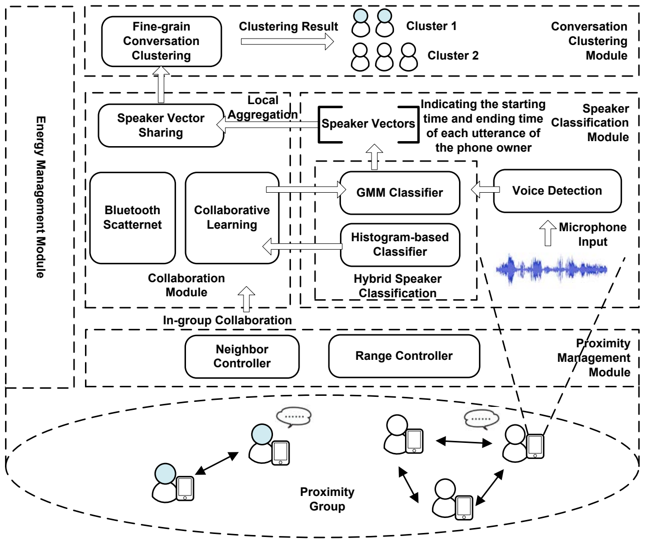


Figure 1: System Architecture of SocialWeaver

if the topology remains fixed. However, the performance would also be affected if the speaker topology changes frequently. SocialWeaver reduces the identification effort by performing speaker classification to detect the voice samples of the phone owner only. The hybrid speaker classification approach used in SocialWeaver is similar to Darwin[19], where collaboration among phones is exploited for speaker model sharing and speaker inference. However, while Darwin requires the exchange of many segments in the collaborative inference process, SocialWeaver exploits collaborative verification only in the model learning phase to acquire high quality training samples to train the speaker models. We will show in the evaluation section that the speaker classification module in SocialWeaver incurs substantially less data transfer.

Conversation Group Detection. Automatic conversation detection is still an active research area due to its importance as well as complexity. [6] exploits cross-correlation between separate audio streams to measure energy synchronicity and use this information to detect conversations. Mutual Information or MI is used in [2, 5, 30] to capture the synchronicity of human voice signals from all users to decide whether they are in the same collocation group. If two persons are both "active" and "collocated", they are classified as being in the same conversation group. However, these approaches consider all users in the same proximity and do not differentiate conservation groups that are physically close to one another. SocialWeaver is unique in that it detects fine-grain conversations by performing a second layer conversation clustering based on the communication patterns on top of the acoustic proximity group.

Conversation Pattern Analysis. Internal conversation pattern is one of the most important context information for interaction-aware applications and has been explored by many researchers [2, 23]. SocioPhone [15] captures meta-linguistic contexts of conversation such as turn-taking. While conversation pattern analysis is important, SocialWeaver focuses on another important aspect of human conversation research, i.e., conversation group clustering and network construction.

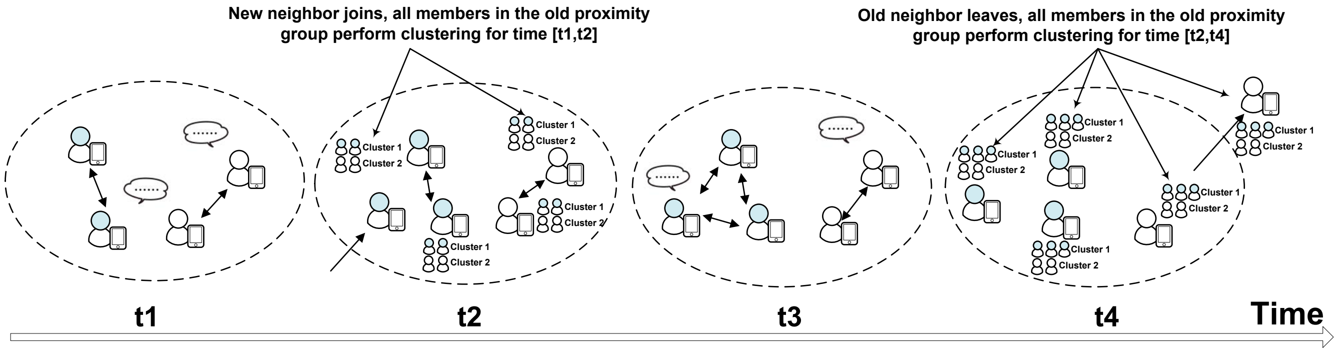

New neighbor joins, all members in the old proximity group perform clustering for time [t1,t2]
Old neighbor leaves, all members in the old proximity group perform clustering for time [t2,t4]
Cluster 1
Cluster 2
t1
t2
t3
t4
Time

Figure 2: Workflow of SocialWeaver

3. OVERALL DESIGN OF SOCIALWEAVER

As shown in Figure 1, SocialWeaver is made up of five modules. We first give a brief overview of each component in this section. Detail descriptions of these modules will be presented in the next few sections.

Proximity Grouping. This module performs the first level of filtering for conversation detection. SocialWeaver relies on the relatively short range of Bluetooth for proximity grouping. All members in the proximity group are assumed to be close enough to converse. Conversely, users holding phones that are beyond the Bluetooth range are assumed to be not in conversation. All further collaboration and conversation detection are based on the proximity groups formed.

Speaker Classification. This is a major challenge in this work. We combine different speaker classification algorithms to adapt to different environments and to minimize the need for pre-training. The hybrid speaker classification scheme uses a dynamic energy histogram-based classifier at the initial phase when a new proximity group is detected and a GMM classifier is used once sufficient samples are collected. In order to train the *in situ* speaker models, we incorporate a collaborative learning module that enables different neighboring phones to share information and collect training samples automatically. In SocialWeaver, the speaker classification module determines whether a voice segment belongs to the phone owner and generates speaker vectors to represent the speaking history of the phone owner for further conversation clustering. A speaker vector indicates the starting time and ending time of one utterance from one user in the proximity group.

In-group Collaboration. Collaboration among users in the proximity group is exploited to both improve speaker classification accuracy and share speaker vectors for conversation clustering. Information exchange in SocialWeaver is based on the Bluetooth scatternet formed in the proximity group.

Fine-grain Clustering. SocialWeaver does not store the audio samples, but instead captures a user's speaking duration. By sharing these time-stamps with other users, fine-grain clustering can be obtained by looking at the temporal characteristics of others users in the same proximity group.

Energy Management. Resource constrain is one of the most challenging problems, and one need to carefully manage the energy consumption of sensing application in order not to affect user's daily phone usage. SocialWeaver builds an energy management module based on POMDP to adapt to different environments.

System Workflow. SocialWeaver runs continuously in the background. As shown in Figure 2, when a new neighbor joins the proximity group at t_2, a new clustering process is triggered. Each phone re-computes the clusters based on the aggregated speaker vectors available between t_1 and t_2. At t_4, when one neighbor leaves the proximity group, a new clustering process is again performed based on the speaker vectors available between the time interval t_2 and t_4. In this way, each phone maintains an evolving conversation network that continuously captures all conversations that have happened within its proximity range. By aggregating conversations over time, conversation network is built for all users.

4. PROXIMITY MANAGEMENT MODULE

The basic function of the proximity management module is to decide if devices are "physically close". There are two related issues. First, how to determine if two devices are within proximity range. Second, how the proximity decision can be utilized.

In SocialWeaver, Bluetooth provides identity, proximity and networking. Two devices are considered to be in proximity if they can discover each other in the Bluetooth neighbor discovery process. Bluetooth is chosen because it is widely available on most smartphones and is energy efficient compared to other interfaces such as WiFi or the use of GPS sensor. We define a proximity group as the set of devices that are neighbors on the Bluetooth network.

Whenever a new Bluetooth device is detected by the phone, a new neighbor entry will be created for this device. To avoid fluctuation and frequent trigger for clustering process, a TTL (*Time-To-Live*) value is initialized for each new entry. Neighbor discovery is periodically performed by each phone, and in each round the TTL will either be decreased by one or refreshed to the initial value if the corresponding device is found again. Once the TTL reaches zero, the entry will be deleted from the neighbor list and the neighbor is no longer in the proximity group.

The maximum distance between two devices in the same proximity group depends on the application scenario. For example, applications capturing discussions in a conference room have much bigger conversation range ($\sim$10m) than applications analyzing conversations in different office cubicles ($\sim$3m). In order to allow the proximity range to be more

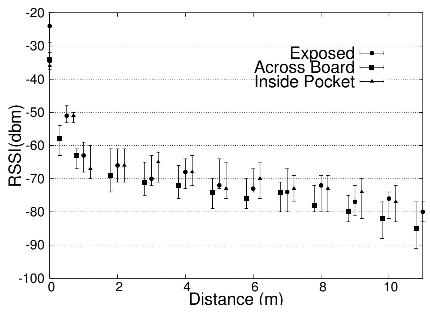


Figure 3: Bluetooth RSSI v.s. Distance in Different Environments Measured by Samsung Galaxy S2 and Samsung Galaxy Nexus GT-I9250

adaptive with respect to the environment, SocialWeaver controls proximity range by using Bluetooth RSSI values as a crude ranging mechanism. As shown in Figure 3, our experiment shows that Bluetooth RSSI is a reasonable indicator of neighbor distance in three different environments, when the phones are exposed, put in pockets or separated by office cardboard partition.

In SocialWeaver, it is assumed that if two persons are in a conversation, their phones must be within the Bluetooth discovery range. Though RSSI values vary depending on various environmental factors, we can always use a conservative (lower) threshold to make sure that nearby neighbors can be detected with high likelihood. We set the RSSI threshold to -90dBm, which covers a range of up to 11m in the environments measured.

5. SPEAKER CLASSIFICATION MODULE

The speaker classification module takes the microphone signal as input and determines if the given voice segment belongs to the current phone user. The first step is to determine if an audio segment recorded is a voice segment. After a voice segment is detected and extracted, a hybrid classification scheme containing two classifiers are used to do the user/background classification for each voice segment to decide whether the voice belongs to the current phone user or other background users in the same proximity group.

VAD (Voice Activity Detection) is applied to each segment of raw audio data to filter out non-voice inputs. Short-term Energy, Zero-Crossing-Rate and Spectral Entropy are used. All these features are widely used in the automatic voice detection systems and have been proved to be light-weighted and efficient [16, 21, 24].

A simple threshold-based algorithm is used on these features and a smoothing window containing five segments (about 320ms when sampling at 16 kHz) is applied to smooth the voicing result. Although the accuracy of the algorithm is not as high as some more sophisticated VAD algorithms in noisy environments [28], it provides sufficiently high recall rate and incurs low computation overheads. In the rest of this section, we present the details of the two classifiers used in the hybrid classification scheme.

5.1 Histogram-based Classifier

SocialWeaver assumes no prior knowledge about any speaker. To perform user/background classification, SocialWeaver first uses a classifier solely based on historical energy information. The assumption is that the voice of the phone owner is usually louder than the voices recorded from other users if averaged over time. While this assumption is only true on average, it greatly depends on the phone placement and voice of the speaker. However, it provides useful information for raw classification when no speaker model is available at the beginning and is a useful tool to bootstrap more sophisticate speaker models as we will show later.

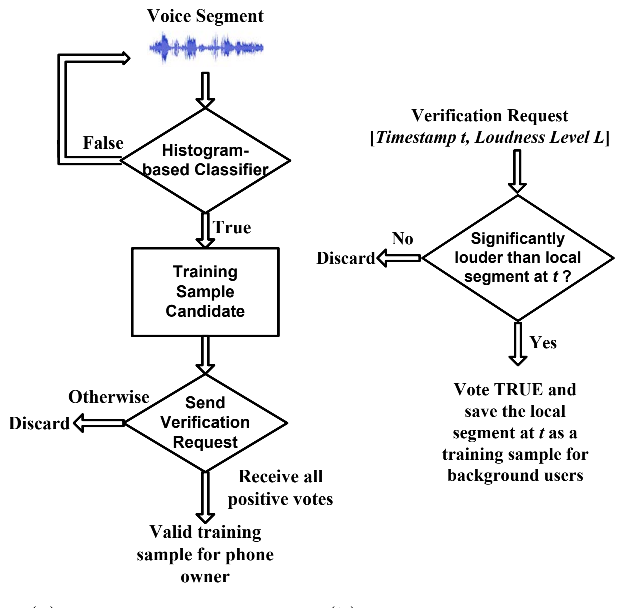
Loudness Adaptation Since different hardware have different sensitivities, SocialWeaver uses normalized loudness instead of traditional RMS (root mean square) energy taken on the raw audio waveform to minimize the effect of differences in phone hardware. Given the audio signal input M which contains k samples, SocialWeaver first calculates the RMS energy and normalizes it to the range [0,1] by letting E $= \frac{\sqrt{\sum_{i=1}^{k}(M_i)^2/k}}{E_{max}}$, where E_{max} is the maximum RMS value possible. Starting from the initialized histogram H_0, a histogram is then built to capture the energy distribution of all voice inputs using E. Using the histogram available, the probability distribution function $p(x)$ of each normalized energy level can be obtained. We define *loudness level* of voice inputs as $L(E_i) = \sum_{x=0}^{E_i} p(x)$, where E_i is the normalized energy of the i_{th} voice input. As an example, 70% loudness level means that the current voice energy belongs to the top 30% in the current energy distribution.

By using the concept of normalized histogram loudness level instead of actual energy measurements, SocialWeaver is able to work better across different phone hardware. The histogram is updated periodically and an aging factor of a, $0 < a < 1$, is performed periodically every 1000 audio segments (about 1 minute when sampling at 16 kHz). The aging mechanism enables the system to adapt to the environment gradually.

A voice segment is classified as belonging to the phone's owner if its loudness level exceeds the threshold $Thresh_{abs}$. Otherwise, it is classified as belonging to the other users. This threshold value affects the precision as well as the recall of the histogram-based classifier.

Environment Adaptation. Another issue that needs to be addressed is the change of smartphone placement. For example, users can move their phones from the desk to pocket/backpack and vice versa frequently, and the loudness histogram varies significantly when phones are placed in different environments. While the aging process in the histogram update provides some form of adaptation, we add an additional mechanism to speed up the adaptation process.

Two common environments that have significant impact on the loudness histogram are *Exposed* (e.g., holding the phone, on the desk) and *Not-Exposed* (e.g., in the pocket or backpack). SocialWeaver maintains a state machine containing two states (Exposed and Not-Exposed). The state transition is triggered by changes in the light sensor readings on the smartphone. In order to filter out spurious transition, state change occurs only if the light intensity values remains relatively stable over a period of a few seconds. Before a state transition occurred, the current loudness histogram is saved and the last known histogram in the new state is reused rather than starting the loudness measurement without memory. We evaluate the energy histogram performance with and without dynamic adaptation in Section 9.


Voice Segment
False
Histogram-based Classifier
True
Training Sample Candidate
Otherwise
Discard
Send Verification Request
Receive all positive votes
Valid training sample for phone owner
Verification Request [Timestamp t, Loudness Level L]
No
Discard
Significantly louder than local segment at t ?
Yes
Vote TRUE and save the local segment at t as a training sample for background users

(a) Verification Requester (b) All Neighboring Phones

Figure 4: Collaborative Verification Process for Automatic Speaker Model Learning

5.2 Probabilistic Classifier

It is well known that the key to accurate speaker identification is to characterize the speaker using speaker-dependent features and build a discriminative model which can effectively distinguish the speaker from all other background speakers. One of the most commonly used features in speaker identification is Mel-Frequency Cepstrum Coefficient (MFCC) [8]. MFCCs are usually extracted from the training samples to train the speaker model for future identification. Gaussian Mixture Model (GMM) has been widely used and proven to be effective in text-independent speaker identification systems [26].

As the histogram-based classification is affected by the environmental factors, using it only is not sufficient. We improve the accuracy of speaker classification by incorporating GMM classifiers. One of the design objectives of our system is that there is no requirement for the availability of training samples of any speaker in advance. Our approach is to automatically verify and train all the models through collaboration among the smartphones.

Collaborative Learning. SocialWeaver trains two GMM models for speaker classification, one for the current phone user and one for background users. Background users consist of all other neighbors in the proximity group. Once a voice segment is accepted by the histogram-based classifier, it becomes a candidate for the next level of validation. In this next step, SocialWeaver exploits a voting mechanism among all neighbors in the proximity group to verify the validity of training samples.

As shown in Figure 4, whenever a voice segment is labeled as a training candidate by the histogram-based classifier, a verification request is broadcast to all neighbors. The request contains two parts, a *timestamp*[1] of the voice segment, and *loudness level* indicates the (normalized) loudness of the voice segment to the phone. Since all neighbors in the same proximity group are physically close to each other in the same environment, it is unlikely that one phone belonging to the non-speaking user receives voice samples that are significantly louder than all the rest of the phones. On the contrary, if one receives such a signature, we have high confidence that the voice segment belongs to that phone's owner. This signature is much more reliable than using only the local histogram. Each phone votes based on local observation and remote loudness value using relative ratio and $Thresh_{rel}$. Once a phone receives positive votes from all neighbors, the requester saves the voice segment as a training sample for the phone's user, and all other phones save their local samples as background training samples.

[1]We assume the phones are synchronized at least to the granularity of one second.

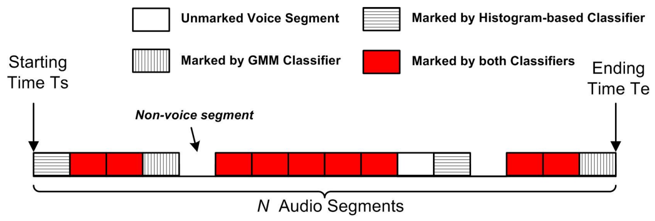


Figure 5: Hybrid Classification Window

Probabilistic Speaker Classification. After phone i collects enough samples for training both speaker and background, voice features of all training samples will be extracted and a speaker GMM model λ_s^i and background GMM model λ_b^i with 32 components will be trained using the EM algorithm [26]. Based on the same assumption as in [26] that the probability of speaking for all users in the proximity group are the same, then $Pr(\lambda_b^i) = N_b \cdot Pr(\lambda_s^i)$, in which N_b is the total number of neighbors in the proximity group. For a given voice feature X captured by phone i, the classifier outputs:

$$\hat{S}_i = \begin{cases} Speaker, & if\ Pr(X|\lambda_s^i) > N_b \cdot Pr(X|\lambda_b^i) \\ & and\ Pr(X|\lambda_s^i) > p_0 \\ Background, & otherwise \end{cases} \quad (1)$$

where $Pr(X|\lambda_s^i)$ is the GMM likelihood of feature X in model λ_s^i and p_0 is the minimum likelihood accepted.

5.3 Hybrid Speaker Classification

In each classification round, a window with N segments are maintained. For each voice segment detected, the two classifiers work independently to decide if the voice belongs to the speaker or background neighbors. Each voice segment is classified as either belonging to the speaker or background users. At the end of the classification window, SocialWeaver computes the speaker score as:

$$S_{hybrid} = w_e \cdot N_e + w_p \cdot N_p \quad (2)$$

where w_e, w_p are the weight of histogram-based classifier and probabilistic classifier respectively and $w_e + w_p = 1$. N_e, N_p are the number of segments in this window marked by each classifier. SocialWeaver decides that current phone user speaks during the window if $S_{hybrid} > c_h \cdot N$, where c_h is the classification coefficient controlling the acceptance of voice segments. We use a classification window with size

$N = 15$ (approximately 1 second) and $c_h = 0.5$ to accept the current window if half of the slots are marked.

Once the voice segments in a window are accepted as speaker utterance, a *speaker vector* $[T_s, T_e]$ will be generated to indicate that the phone user speaks from time T_s to T_e.

6. COLLABORATION MODULE

A unique feature of our system is the use of multi-hop network communications for collaborations among the smartphones. Collaboration among users is used to accomplish tasks such as time synchronization, collaborative learning and speaker vector aggregation.

6.1 Communication Topology Formation

SocialWeaver exploits Bluetooth for neighbor discovery. Bluetooth uses a master/slave structure. To form an *ad hoc* network with interconnected Bluetooth devices, Bluetooth devices form a *piconet*, where the master can have up to 7 slaves. To scale the Bluetooth *ad hoc* network, independent piconets can be connected to form a *scatternet* [1, 29]. All nodes in the scatternet are interconnected either through direct link connection or indirect multi-hop connections.

Whenever a message needs to broadcast from one node to others in SocialWeaver, a flooding mechanism is employed. Each new message is given a randomly generated 32-bit sequence number by the sending node. All receiving nodes receive the message and check if the message with the same sequence number has been processed recently. If a message is new, it will be forwarded to all connected nodes. Otherwise, it is discarded.

6.2 Synchronization

Since synchronizing from cellular network or GPS depends on user phone settings, SocialWeaver uses a decentralized scheme for coarse time synchronization. Each phone maintains a time offset table for all neighbors. Periodically, a *Timesync* message with local timestamp will be generated by each phone and broadcast to all neighbors, and all phones receiving the message computes the local and remote time difference and updates the offset for that neighbor. Since the message latency ranges from several milliseconds to several hundred milliseconds within the scatternet, the time offset calculated contains additional delays and is not very accurate.[2] However, since the inaccuracy is in the order of seconds or less, it is not large enough to significantly affect the collaborative learning and conversation clustering results. The simplicity of this synchronization scheme satisfies our system requirements.

7. CONVERSATION CLUSTERING MODULE

In this section, we will show how the speaker vectors are used for conversation clustering. First, speaker vectors generated locally are aggregated to form vectors of longer duration to improve temporal coverage and reduce overhead. Next, these locally generated and aggregated vectors are shared among all devices in the proximity group (using Bluetooth) and conversation scores are computed for different pairs of speaker vector. Finally, based on the conservation scores, conversation clustering is performed.

[2]To improve accuracy, we ignore the largest 10% of the values and computes the average offset with the rest of the samples.

7.1 Speaker Vector Aggregation and Sharing

A speaker vector generated by one phone represents the starting time and ending time of one speech segment from the phone owner. As these speaker vectors are to be shared with other users, it is important to reduce the network traffic. *Local aggregation* is performed to merge the adjacent speaker vectors. As shown in Figure 6(a), 3 speaker vectors of the phone's owner that are temporally close together are merged into a single vector of longer duration. These aggregated vectors are broadcast to neighboring devices periodically. The sharing of speaker vectors enables each phone to have sufficient information to describe the conversations.

7.2 Conversation Score

To measure how likely two persons are involved in the same conversation, Mutual Information (MI) has been used as a measure of the synchronicity of two audio inputs [2]. If the two voice signals are correlated, that is, they are either perfectly aligned or perfectly misaligned, the MI is high. MI is useful for discovering physically collocated people whose voice inputs are highly correlated. However, MI alone based on binary voice signals provides no strong evidence of whether two collocated people are in the same conversations group. To measure the conversational correlation between a pair of speakers in the same proximity group, we propose a new metric called *Conversation Score*:

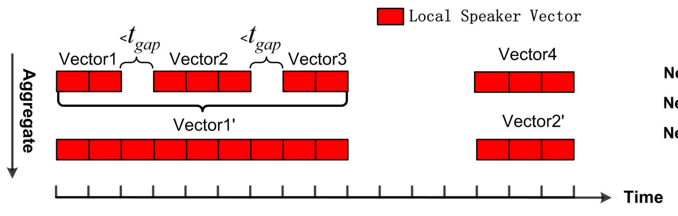
$$CS_{ij} = p_r - \alpha p_c - \beta p_s \tag{3}$$

where CS_{ij} is the conversation score of user i and j. First, users i and j are in the same proximity group. p_r is the ratio of time that when user i is speaking, user j is not, or vice versa. p_c is the ratio of time that both users are speaking. Finally, p_s is the ratio of time where both users are silenced. The intuition is as follow. If two persons are involved in the same conversation, the speaking pattern should be highly synchronized. If at each slot there is only one speaker speaking then a reward is added to support the observation that they are having a conversation. On the other hand, if both speakers speak at the same time or are mostly silent, they are less likely to be in the same conversation. α and β are used to control the collision penalty and silence penalty respectively.

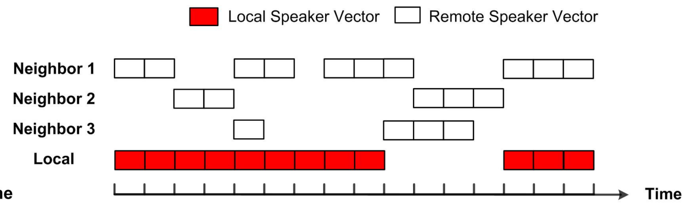
We illustrate the calculation conservation score using Figure 6(b). There are 16 timeslots and the nodes are in the same proximity group. For devices 1 and 2, $\{p_r$=0.94, p_c=0, p_s=0.06$\}$, and for device 1 and local user, $\{p_r$=0.25, p_c=0.56, p_s=0.19$\}$. Based on the speaking pattern, it should be obvious that users 1 and 2 are much more likely to be having a conversation than user 1 and the local user. Evaluations are presented in Section 9.

7.3 Conversation Clustering

Conversation score measures the likelihood of conversation between 2 speakers. SocialWeaver extends pairwise conversation to arbitrary conversation group size using a conversation clustering algorithm. We assume that each member in a proximity group can be involved in only one conversation group. And there may exist several conversation groups within one proximity group. Each of them is made of disjoint set of users. The conversation clustering



(a) Local Aggregation



(b) Global Sharing

Figure 6: Speaker Vector: Local Aggregation and Global Sharing

in SocialWeaver finds these disjoint clusters in two phases: *split* and *merge*.

(1) Split. In this step, the proximity group is split into disjoint initial clusters S which consist of only one single user per cluster based on the following rule:

$$\begin{cases} \underset{S}{maximize}\, \|S\| \\ s.t.\ \forall i \in S\, \forall j \in S\, (CS_{ij} < 0) \end{cases} \tag{4}$$

In this step, we form as many initial clusters as possible, such that nodes from any two initial clusters are very unlikely to be in conversation based on conversation scores. All nodes in the initial clusters become cluster heads for future merge operations, and each pair of cluster heads are in different conversation groups. After each pairwise conversation score is computed, a conversation score matrix is obtained. Based on the matrix, we form a graph with n nodes (number of members in the proximity group) and each pair of nodes are connected if their conversation score is greater than 0. Solving this problem is equivalent to finding the maximum independent set from the graph. Although the optimization problem is NP-hard, the number of speakers in one proximity group is usually small and independent sets can always be found sufficiently fast. If we set $\alpha = 10$, $\beta = 0.1$, one maximum independent set found for the split process for Figure 6(b) is $S = \{Local, 1\}$.

(2) Merge. After the initial split phase, let there be $k = \|S\|$ initial clusters. To assign the rest of the devices into one of these clusters, we solve the following optimization problem:

$$\arg\max_{S'} \sum_{i=1}^{k} \sum_{j \in S'_i} CS_{ij} \tag{5}$$

in which CS_{ij} is the conversation score between user j and the initial member of the cluster S'_i. We therefore find a clustering $S' = \{S'_1, S'_2, ..., S'_k\}$ that maximizes the total conversation score of the system. A simple $O(n^2)$brute-force algorithm is used to assign each node to the set with largest conversation scores. Since the number of members in a proximity group is usually small, it would not affect the system performance. Following the merge process, the example shown in Figure 6(b) reduces to two conversation groups $S' = \{(Local, 3), (1, 2)\}$.

8. ENERGY CONTROL

As smartphone is a resource-constrain device, energy efficiency is a major concern. To achieve energy efficiency, several approaches have been proposed, including triggered-sensing [22], code offloading [7], hardware support [17] and dynamic duty cycling [18]. The MDP-based duty cycling proposed by Jigsaw [18] leverages the fully observability of each state to decide the next GPS sampling rate to perform. In SocialWeaver, however, the system state is not fully observable to the phone. Hence, we exploit Partially Observable Markov Decision Process (POMDP)[10] to model the uncertainty of human factor (voice activity) to extend the approach used in [18].

A POMDP model can be described with the tuple (S, A, O, T, Ω, R), where S, A, O are the finite set of states, actions and observations respectively. At each discrete time step t, one action A_t is taken, on which the state changes from S_t to S_{t+1}. The agent cannot observe the current state of the environment but only updates the belief, which is the distribution of states, based on the sensory input. The agent receives rewards for each action performed in a state. The POMDP is characterized by state transition function $T = p(S_{t+1}|S_t, A_t)$, observation function $\Omega = p(O_t|S_t, A_t)$ and reward function $R(S_t, A_t)$. The goal is to find out a control policy π_γ that that maps current belief to actions that maximize the expectation of sum of rewards, i.e.,

$$\arg\max_{\pi_\gamma} E \sum_{t=0}^{\infty} \gamma^t R(S_t, A_t) \tag{6}$$

where $\gamma \in (0, 1)$ is a discount factor to ensure convergence of the model.

In SocialWeaver, each state at time step t is represented as $S_t = (E_t, V_t, P_t)$. $E_t \in \{0, 1, 2, ..., 100\}$ is the current percent of energy left measured by the phone. $P_t \in \{0, 1\}$ is the proximity level, where 0 indicates there are no phones nearby and 1 otherwise. Both E_t and P_t are fully observable to the phone. V_t describes the current voicing level of human members in the proximity group, i.e., the percentage of the time in timeslot t that people in the proximity group are speaking. In this model, V_t is the only variable that is not directly observable. We divide the voicing status into 6 voicing state: {0,1,...,5}, where 0 means no voice event at all, 1 means 20% voice activity, and so on.

SocialWeaver uses Bluetooth for proximity detection and microphone for voice and speaker detection. We combine Bluetooth and microphone in the action space, i.e., $A_t = (B_t, M_t)$. For Bluetooth we use two different modes, {*0:idle, 1:connect*}. Idle mode uses a low duty cycling rate to save energy and detect new proximity members, while connect mode establishes scatternet with nearby phones for speaker vector exchange. The microphone M has 6 different duty cycles, {0,1,...,5}, representing percentage of time microphone samples at each time slot. Hence, 0 means no sampling, 1 means to sample 20% of the time and so on.

The state transition probabilities $p(E_{t+1}|E_t, A_t)$, $p(P_{t+1}|P_t)$ and $p(V_{t+1}|V_t, P_t)$ are learned from the traces. The observation is the percentage of time that voice is detected at each time slot. The probability can be approximated by calculating the overlapping between the real voice interval and microphone sampling interval, therefore,

$$p(O_t|S_t, A_t) = p(O_t|V_t, M_t) = \frac{\binom{V_t}{O_t}\binom{5-V_t}{M_t-O_t}}{\binom{5}{M_t}} \quad (7)$$

The reward function in our model is defined as:

$$R(S_t, A_t) = \begin{cases} f_m - f_p, & E_t \neq 0 \\ 0, & E_t = 0 \end{cases} \quad (8)$$

where $f_m = M_t \cdot V_t + B_t \cdot P_t$ and $f_p = \frac{c_1 \cdot M_t^2 + c_2 \cdot B_t^2}{E_t}$.

f_m and f_p reflect the rewards for using high and low duty cycles respectively. c_1 and c_2 are empirically determined coefficients that adjust the weight of Bluetooth and microphone actions. When the energy left is large, f_m dominates, and the phone will be encouraged to use higher duty cycling rate to increase the reward. However, when energy becomes scarce, f_p becomes more important and a better policy needs to take V_t into account and only use high sample rate when V_t is high. The reward function adapts the policy to environment and current phone status to increase the coverage of conversations and accuracy of detection.

We use the SARSOP algorithm [12] to compute the policy. The action determination is light-weight and can be performed in real-time.

9. EVALUATION OF SOCIALWEAVER

In this section, we evaluate the performance of the various components of SocialWeaver and the overall system performance through small scale user studies.

Implementation. SocialWeaver is implemented on the Android platform (version 2.3.6 and above). It runs as an Android service in the background and has been tested on Samsung Galaxy S2, Samsung Galaxy Nexus GT-I9250 and HTC Desire phones. The code for signal processing, collaboration, learning and conversation clustering are about 3000 lines of Java code. SocialWeaver uses 16 kHz sampling rate for microphone and the neighbor discovery using Bluetooth is performed periodically by default.

Dataset. We evaluate SocialWeaver through both controlled experiments where the interaction patterns can be easily verified, and uncontrolled where people behave naturally. We have collected data from different environments, including group meeting in relatively quiet meeting rooms or research laboratories, and social events such as lunch and dinner in cafeteria. In the controlled experiments where we evaluate the performance of the speaker classification module, raw audio inputs are collected and manual tagging is used to establish the ground-truth. Above 100 hours of raw audio containing real conversation data are collected and analyzed. To evaluate the overall system performance, a 5-day controlled experiment involving 10 participants and an uncontrolled, 1-hour classroom teaching event involving 13 participants are used. As only logs and no raw audio sample are collected for these experiments, the interaction patterns are noted for analysis.

9.1 Speaker Classification

We evaluate the performance of SocialWeaver's speaker classification module through two metrics, namely precision and recall. Precision is the percentage of the classifier's positive output containing true owner's voice and recall is the percentage of owner's voice detected from the entire audio stream. There are other metrics, such as Diarization Error Rate (DER), that are widely used in speaker diarization systems. We choose to use precision and recall so that we can have interpretation with respect to and direct comparison to similar systems such as those used in Darwin[19] and SpeakerSense[17]. We evaluate the performance in a noisy outdoor and a quiet indoor environment. For baseline, we compare with the speaker identification algorithm used in Darwin.

Scenario 1: Noisy Outdoor Environment. The outdoor experiment was performed in the campus cafeteria. Participants sat close to each other (less than 2 meters) most of the time. In total, 50 hours of conversations are collected. We present the results in stages, showing the pros and cons of the individual technique and finally the actual algorithm.

In Figure 7, we show the precisions and recalls for classification using only histogram-based classifier. The precision increases from 60% to 70% as the absolute threshold for energy histogram increases. As expected, the recall drops accordingly as $Threash_{abs}$ increases as the classifier becomes stricter. The low precision of the histogram-based classifier is due to the high cross-picking from nearby speakers and noisy environment. A histogram-based only classifier is clearly inadequate.

The first improvement is to introduce the use of environment adaptation through the light sensor as mentioned in Section 5.1. Figure 8 shows the result. $Thresh_{abs}$ is set to 0.5, and the phone is moved from table to pocket at the 30 minute mark. Without dynamic adaptation, recall drops as the histogram built previously is insensitive to low energy voices but voice energy got significantly reduced after the phone is put into the pocket, resulting an increase in false negative. In the second case, the phone is moved from pocket to table. Figure 9 shows a significant drop in the precision when the phone is taken from pocket to the table without dynamic adaption. This is because the histogram built has been customized to low energy environment and tends to accept almost all voices when it is placed on an "exposed" environment, introducing lots of false positives and also increase in the recall.

The performance of histogram-based classifier (plus environment adaptation) is still not sufficiently accurate. The performance can be improved substantially by incorporating collaboration. Figure 10 shows the result with collaborative verification included, with $Threash_{abs} = 0.5$. With collaboration, precision improves to 90%, while recall drops accordingly. However, since this high precision data set is collected automatically, it can be used as a set of high quality training samples for the probabilistic based classifiers to further boost the performance.

Figures 11 and 12 show the result of GMM classifier as the size of training sample increases with and without collaborative learning respectively. The performance of the classifier becomes stable after the training size becomes larger than 30 seconds in both cases. Without collaboration and using histogram only, precision of GMM classifier is only about

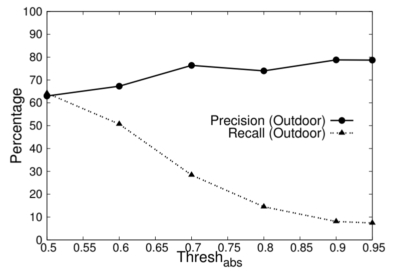


Figure 7: Histogram-based Classifier (Outdoor)

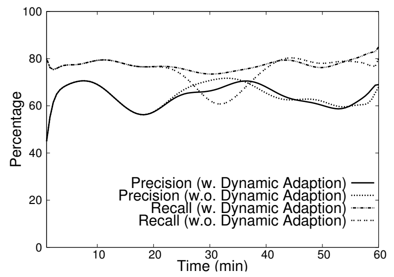


Figure 8: Environment Adaption: Table to Pocket

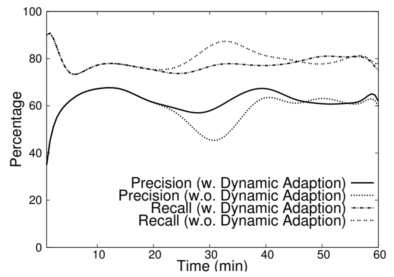


Figure 9: Environment Adaption: Pocket to Table

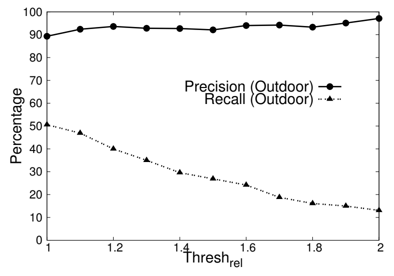


Figure 10: Collaborative Verification (Outdoor)

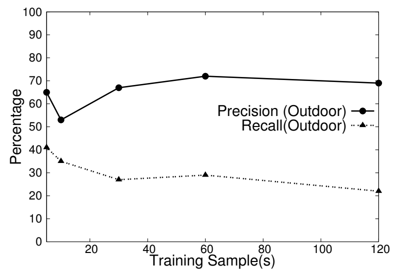


Figure 11: GMM Classifier without Collaboration (Outdoor)

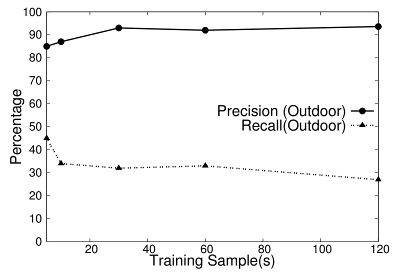


Figure 12: GMM Classifier with Collaboration (Outdoor)

70%. Once collaboration is included, precision becomes as high as 90%. However, the recall is rather low, only 30% to 40%. In any case, we successfully exploit the non-obtrusive and ubiquitous histogram-based classifier to build an *in situ* acoustic feature based classifier.

Finally, by combining the relatively high recall of the histogram-based classifier and high precision of GMM classifier, the hybrid classification scheme achieves a more reliable performance. As shown in Figure 13, the precision of histogram-based classifier only ($w_e = 1, w_p = 0$) can get as low as 60% due to cross-picking in the noisy and dense speaker environment. GMM classifier ($w_e = 0, w_p = 1$) is more robust but the recall is still low. The hybrid classifier ($w_e = 0.5, w_p = 0.5$) improves the precision of histogram-based quality classifiers by using a second layer GMM classifier based on the voice features of the speaker and reduces false positives effectively, achieving a high precision. On the other hand, the true speaker voice (true positives) passing the histogram-based classifier tends to pass the GMM classifiers as well, resulting a recall boost for the GMM classifier. Overall, the hybrid classifier can effectively detect phone user's voice with precision up to 85% and 60% recall for this outdoor scenario.

As a baseline, we compare our scheme to the Darwin system. Figure 14 shows the inference performance of Darwin and SocialWeaver over 10 minutes. We can see that both systems achieve comparable performance, with the Darwin system achieving slightly higher precision but lower recall. This is because Darwin is stricter in its inference. On the other hand, SocialWeaver incurs much less communication overhead. As shown in Figure 15, SocialWeaver finishes collaborative learning process within the first minute. A total of 20 verification requests and 10 responses are sent. After the learning phase, all inference is done locally and no further communication is required for speaker inference. On the other hand, Darwin continues to communicate as each inference has to be agreed by all participants.

Scenario 2: Quite Indoor Environment. In this setting, 13 participants sat in conference room and the distance between two participants varies from 1m to 10m. 50 hours of conversations over two months are collected. As shown in Figure 16, precision remains as high as 90% in the indoor environment. This is not surprising since cross-picking between different speakers in this environment is small and the histogram-based classifier performs well.

Collaborative learning further boosts the precision of speaker inference. The precision of results collected through collaborative verification indoor goes above 90% as expected, in the sacrifice of lower recall. Figure 17 shows the GMM performance with collaborative verification, achieving about 90% precision and 40% recall. Figure 19 shows the hybrid classifier aggregating both classifiers. The hybrid classifier compliments both classifiers, achieving 90% precision and 70% recall in this scenario.

Figure 18 shows the comparison between Darwin and SocialWeaver in the indoor environment. Both systems achieve about 90% precision, with SocialWeaver having slightly higher recall. In this case, speaker talks less and it takes two minutes for SocialWeaver to learn the model. The result for communication overhead is similar to the outdoor scenario and is not shown.

9.2 Collaboration Module

Scatternet Formation. The communication in SocialWeaver among neighboring phones is based on the Bluetooth *ad hoc* networking. Figure 20 shows the delay for scatternet formation as the number of phones increases. The delay increases almost linearly with the number of phones. The delay ranges from 3 seconds to about 2 minutes when the

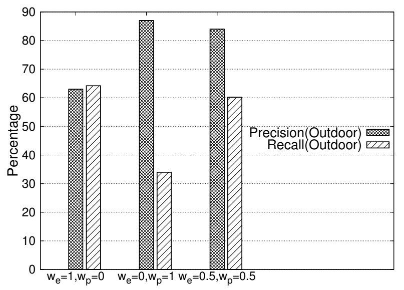


Figure 13: Hybrid Classification with $Thresh_{abs} = 0.5$ (Outdoor)

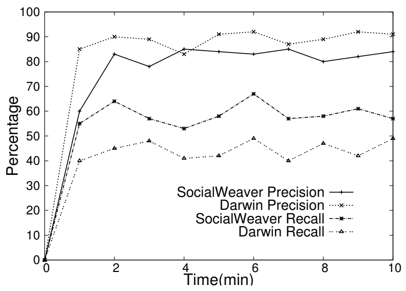


Figure 14: Performance of SocialWeaver and Darwin (Outdoor)

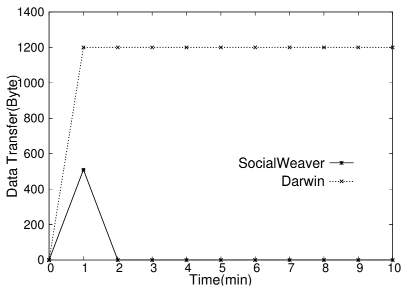


Figure 15: Communication Overhead of SocialWeaver and Darwin (Outdoor)

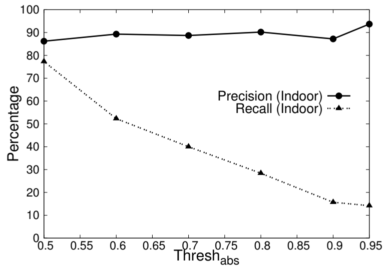


Figure 16: Histogram-based Classifier (Indoor)

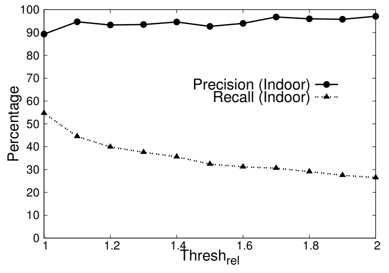


Figure 17: GMM Classifier with Collaboration (Indoor)

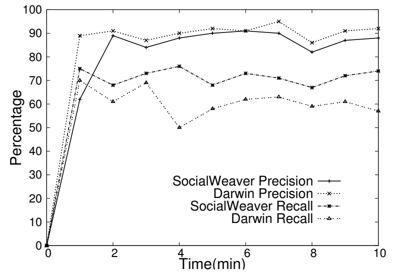


Figure 18: Performance of SocialWeaver and Darwin (Indoor)

number of phones grows from 2 to 16. While 2 minutes may seem long, conversations that involve many users tend to last even longer. Using Bluetooth, our system is more suitable to capture conversations that last for at least a few minutes. If spontaneous conversation shorter than a minute is deemed interesting and important, a different network technology, say WiFi or the more recent Bluetooth 4.0 may be used.

Local Speaker Vector Aggregation. Local aggregation reduces the amount of data exchanged, and for the data set used, the reduction is approximately 25%. The effect of local aggregation on precision and recall is shown in Figure 21. Without local aggregation ($t_{gap} = 0$) all the speaker vectors are generated by hybrid classification module. When $t_{gap} = 2s$, all neighboring speaking vectors with gaps smaller than 2 seconds are merged as one. The recall improves by 15% and 13.5% for dense environment and sparse environment respectively with minimum decrease in precision. The gain in the recall comes from filling the gap between nearby vectors, and the fact that small gaps between two consecutive speaker vectors also belongs to the same speaker with high probability. However, if the merging threshold is set to 5 seconds, the precision start to drop since merging a large gap would introduce more false positives by admitting voices from background speakers. Hence, a $t_{gap} = 2s$ is used for later experiments.

9.3 Conversation Clustering Module

To understand the accuracy for different types of conversations, we divide conversations into four categories: *Long Warm*, *Long Cold*, *Short Warm*, and *Short Cold*. Long conversations are those that last longer than 5 minutes and short conversations last for less than 5 minutes. Warm discussions are those that involve lots of interactions among the speakers. On the other hand, cold discussions are those where speakers talk less and silence dominates. These categories are chosen because the performance of SocialWeaver is strongly influenced by the conversation duration and interaction patterns. Besides clustering accuracy, we also use the widely adopted F1 score as our metric to evaluate the clustering performance. F1 score is defined as $F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$. The closer the F1 score is to 1, the better the clustering result is.

Category	**Internal Conversation Structure**	**Conversation Type**	**Conversation Duration**
1	(0,1,2),(3,4,5)	Long Warm	30 mins×3
2	(0,1,2),(3,4,5)	Long Cold	30 mins×3
3	(0,1,2),(3,4,5)	Short Warm	5 mins×3
4	(0,1,2),(3,4,5)	Short Cold	5 mins×3

Table 1: Dataset for Clustering Evaluation

As listed in Table 1, we conducted controlled experiments where 2 conversation groups exist in each experiment and there are 3 experiments per categories. There are 6 speakers in each experiment, and the first three users (0,1,2) and the next three users (3,4,5) are in the same conversation group.

We measure the performance as follow. A TP (True Positive) decision of the classification assigns two members in the same conversation to the same cluster, and a TN (True Negative) decision assigns two members in two different conversations to two clusters. Therefore, if the clustering generates clusters exactly the same as the ground truth (0,1,2) and (3,4,5), there should be $C_3^2 + C_3^2 = 6$ TPs, and $3 \times 3 = 9$ TNs. The opposite definition works for FP (False Positive) and FN (False Negative). Therefore we have TP+TN+FP+FN=15 for all 12 proximity group clustering. In the evaluation we normalize the value and define accuracy as $(TP+TN)/(TP+$

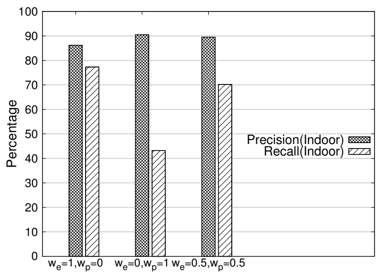


Figure 19: Hybrid Classification with $Thresh_{abs} = 0.5$ (Indoor)

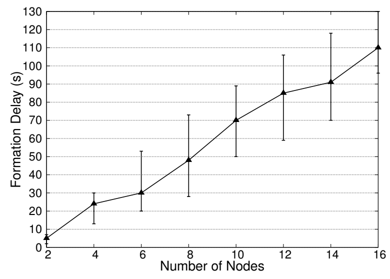


Figure 20: Scatternet Formation Delay

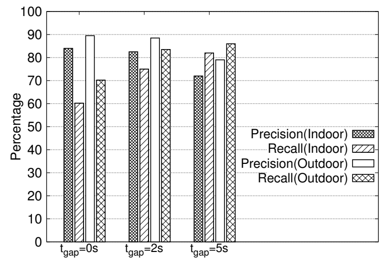


Figure 21: Effect of Local Aggregation on Speaker Classification

Type	**α=5,β=0.01**					**α=10,β=0.01**				
	TP	**FP**	**TN**	**FN**	**Accuracy**	**TP**	**FP**	**TN**	**FN**	**Accuracy**
Long Warm	0.22	0.22	0.38	0.18	0.6	0.22	0.04	0.56	0.18	0.78
Long Cold	0.4	0.6	0	0	0.4	0.36	0.27	0.33	0.04	0.69
Short Warm	0.31	0.13	0.47	0.09	0.78	0.27	0.11	0.49	0.13	0.76
Short Cold	0.31	0.33	0.27	0.09	0.58	0.22	0.22	0.38	0.18	0.6
Type	**α=5,β=0.1**					**α=10,β=0.1**				
	TP	**FP**	**TN**	**FN**	**Accuracy**	**TP**	**FP**	**TN**	**FN**	**Accuracy**
Long Warm	0.31	0.13	0.47	0.09	0.78	0.22	0.04	0.56	0.18	0.78
Long Cold	0.36	0.07	0.53	0.04	0.89	0.36	0.04	0.56	0.04	0.92
Short Warm	0.36	0.07	0.53	0.04	0.89	0.27	0.07	0.53	0.13	0.8
Short Cold	0.31	0.33	0.27	0.09	0.58	0.27	0.16	0.44	0.13	0.71

Figure 22: Conversation Group Clustering Result

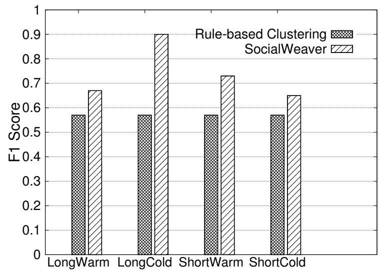


Figure 23: Comparison Between Rule-based Clustering and SocialWeaver

$TN + FP + FN$).

Figure 22 shows the result of conversation group clustering for all 12 proximity groups in different parameter settings. α and β control collision penalty and silence penalty respectively to restrict or relax the clustering result. Overall, as α and β increase, the clustering becomes stricter and false positives decrease while false negatives increase. An interesting observation is that as α increases, the false positive for long warm conversations drops the fastest. This is because in the long warm conversations, the difference of the speaking patterns between two different conversations results in more collisions and are more sensitive to α. Besides, long cold conversations and short warm conversations can also be effectively clustered by controlling α. Increasing the β on the other hand affects long cold conversations the most and reduces the false positives for conversations with long silence. For short cold conversations, however, the conversation only last for a short time and speaker rarely speaks during this period. Neither collision penalty nor silence penalty classifies these conversations effectively. Overall, when $\alpha = 10, \beta = 0.1$, the conversation clustering achieves at least 71% accuracy for all four types of conversations.

Figure 23 shows the comparison between the rule-based method adopted in [30] and SocialWeaver conversation clustering method. In the former, if two persons are in the same group and are active in talking, they will be classified as belonging to the same conversation group. For all different types of conversations, rule-based method achieves the same performance. SocialWeaver, on the other hands, takes communication pattern into account and outperforms rule-based method for all types of conversations.

9.4 Energy Efficiency

In order to evaluate our POMDP policy, we learned the transition probabilities from the weekday, weekend and social event traces collected to simulate the POMDP performance [12]. The performance metrics is Speech Coverage - the percentage of time when voice can be detected when microphone is on. Without POMDP, Bluetooth stays in the connect mode to support collaboration and avoid missing remote speaker vectors. Microphone is sampled using fixed duty cycling 20%, 60% and 100%.

Figure 24 shows one sample execution with POMDP. As expected, Bluetooth switches to connect mode only when neighbors are detected, and microphone tends to be more aggressive at the beginning when remaining energy is large. However, when the battery drains, the phone acts more conservatively and only use higher microphone duty cycles when the observed voice level is high.

Figure 25 shows the average amount of energy left on the phone over time. With fixed duty cycling, either the voice coverage is poor or the battery drains too fast. When the SocialWeaver POMDP policy is enabled, the coverage is over 90% for up to 5 hours, and the battery discharges completely in about 11 hours.

The amount of savings depends on the voice activities to be detected. Figure 26 shows the energy saved in different "load" conditions compared to the cases with energy control disabled. Savings is highest in the Weekend trace, since there is little interaction. While in the social event, there is frequent interaction and the power saved is much less. Overall, our POMDP model works well and saves about 50% on the average over all traces used.

9.5 System Evaluation

We evaluate the overall system performance of SocialWeaver through 2 real-life user studies. The first is a controlled experiment where the interactions among 10 graduate students are tracked over a 5-day duration. The second evaluation was conducted in an actual one hour class pre-

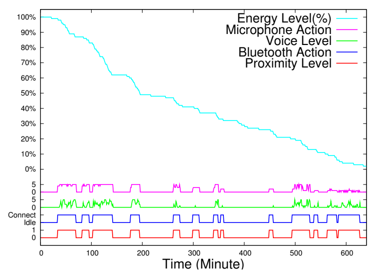


Figure 24: Sample Execution Using POMDP Generated Policy

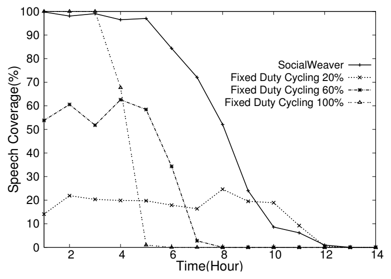


Figure 25: Comparison Between POMDP and Fixed Duty Cycling

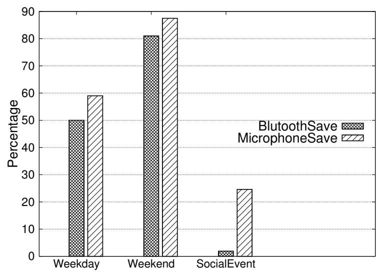


Figure 26: Bluetooth and Microphone Save in Different Scenarios

sentation where different groups of students made a 5-10 minutes presentation.[3]

Controlled Experiment. Over a 5 day period, 10 participants carried the phones with them when they were on campus. 5 participants (ID1∼ID5) belong to the same research group and work in the same lab. The other 5 participant (ID6∼ID10) worked in different labs. 5 participants (ID1∼ID5) met on Monday morning for group discussion and 6 of the participants (ID5∼ID10) are social friends and they met up for lunch and dinner every day. If SocialWeaver generates conversation clusters with only one user involved, we consider this user is having conversation with unknown user and assign this conversation to that user and the dummy user both. A total of more than 500 conversations are detected.

Figure 27 shows the interaction obtained by SocialWeaver. Each day is divided into six 4-hour periods. Darker shade indicates higher conversation intensity.

As expected, there are two peaks for conversations around 12pm and 6pm on all 5 days, when some of the participants (ID5∼ID10) met for lunch and dinner. On Friday night, an event was organized for ID5∼ID10 and conversation intensity is highest from 6pm∼midnight. The other weaker peak is between 10am and 11am on Monday, when some of the participants (ID1∼ID5) met for meeting. At the other times of the day, the interactions among the participants are rare.

Figure 28 shows the conversation network generated by SocialWeaver. The thickness of the edge represents the interaction level between adjacent nodes. As we can see, the conversation network derived from the information collected by SocialWeaver accurately reflects the real world social connections being measured.

Uncontrolled Experiment. In the uncontrolled experiment, we measure the conversation clustering in a classroom setting. There are about 30 students in the class and 10 students participated in the experiment. During class, each group, consisting of 2 or 3 students, gave a 5 to 10 minutes presentation. There are a total of 11 groups and the 10 participants belong to 4 different groups. At the beginning of class, smartphones installed with SocialWeaver were given to the participants. We observed that participants carried the phones in different ways, some placed them on the table and the rest put the phones in shirt or pants pockets. As 3 additional phones were carried by the teaching staff and the authors, there are a total of 13 participants.

While we do not keep track of the identities of the students, based on the experimental setup, we expect the following. (1) Since smartphones installed with SocialWeaver were issued to the participants in groups, we expect the phone identifiers within conversation clusters to be clustered in a similar way; (2) since each group gave a presentation, we expect each group to be the "dominant" or most active speakers at least once; and (3) as ID6 to ID8 are issued to the staff and researchers, we expect them to have high degree of conservation intensity.

Figure 29 shows the conversation clustering obtained. The 1 hour duration is divided into twelve 5-minute intervals. Slots with the same shading indicate same conversation groups and darker shades imply higher conversation intensities. One of the phones (ID9) did not generate any log data, probably because the application was disabled accidentally by the student. The results can be summarized as follows.

(1) Besides {ID6, ID7, ID8}, the clustering shows 4 distinct conservation groups, {ID1, ID2}, {ID3, ID4, ID5}, {ID10, ID11} and {ID12, ID13} which are active over different time periods.

(2) The 4 groups ({ID1, ID2}, {ID3, ID4, ID5}, {ID10, ID11} and {ID12, ID13}) are the dominant speakers at different times, most likely when they were presenting. For example, {ID10, ID11} is the most active during the 0 to 5min interval, {ID13} from 20 to 25min, {ID3, ID4, ID5} from 25 to 30min, and {ID1, ID2} from 35 to 40min.

(3) ID8 is the most active throughout, and is most likely the lecturer.

(4) There can be multiple active conservation groups at the same time. For example, from 0 to 10min, while the group {ID10, ID11} are dominant speakers, there are three other active conservation groups.

(5) Some group continues the discussion after presentation. For example, {ID1, ID2} continue their discussion after they became dominant speakers from 35 to 40min for another 20 minutes till the end of class.

Most conversations in this presentation event belongs to the short warm category. We map the inferred clusters to the presentation schedule and conversation clusters manually tagged by an on-site observer to verify the clustering performance. The accuracy for all conversation clustering shown in Figure 29 is 81.9%, which matches very well what we measured in Section 9.3.

10. DISCUSSION

10.1 Applications Enabled by SocialWeaver

[3]Institutional Review Board (IRB) approval was obtained for this experiment.

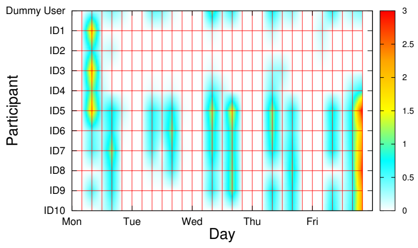


Figure 27: Conversation Clusters Detected in One Week

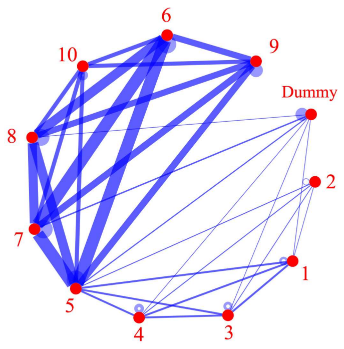


Figure 28: Conversation Network Built Among Participants

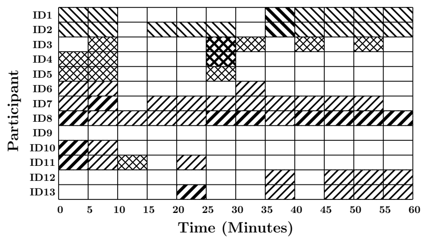


Figure 29: Clustering for 1-hour Presentation Event

As a fundamental service running on mobile phones to automatically capture human conversations and build face-to-face communication networks, SocialWeaver opens up a wide range of new applications.

Communication Topology Analysis. With conversation information extracted by SocialWeaver, the face-to-face communication topology within a community can be easily built. More information can be extracted based on the topology, for example, the social centers in the community who have the most connections to other people and social bottlenecks who separates two networks. Lots of potential information can be mined to improve the efficiency and quality of communication for the community.

Shortest Path Social Extension. As an important functionality of social networking, extending social connections can also benefit from the conversation network. Conversations indicate real-world social connections and are more reliable than online social networks. By looking at the social path from one person to the other, one can find the shortest path to reach the other person in the same community to build up new social connections efficiently.

10.2 Privacy

As data needs to be collected from participants, privacy becomes one of the most challenging problems in people-centric mobile sensing applications. To protect the privacy of users in conversational sensing systems, one key principle is that no raw audio can be saved for future analysis and no verbal information can be recovered from the features collected. Earlier works analyzing human conversations relies on collecting separated streams of inputs. [6] computes correlations between two raw audio inputs and classifies them to the same conversation if the similarity is above the threshold. [2] and [30] detect conversations based on the idea of mutual information computed from separated streams of inputs from all users. Features from raw audios are extracted first before aggregation for conversation detections. SocialWeaver, on the other hand, adopts a novel distributed and real-time computing model. All processing are done locally on each phone, only speaker vectors and loudness level are exchanged for collaboration, no other voice features or raw audio from any user is revealed. This feature of SocialWeaver enables it to be deployed easily while respecting users' privacy.

11. LIMITATION AND FUTURE WORK

In SocialWeaver, collaborative verification is used to complement the histogram-based classification to achieve better performance for speaker classification. However, while collaborative verification provides high quality training samples, recall can be affected by speaker topology settings. While sufficient samples can be collected within minutes in most cases, there are cases where there are insufficient training samples. In such cases, we have to use the less accurate histogram-based classifier. However, when no *a prior* knowledge is available, accurate speaker classification remains a challenge.

For conversation detection and clustering, since SocialWeaver relies on Bluetooth for information exchange, the delay in building the scatternet makes it less likely to captures short, spontaneous conversations. This issue may be addressed through the use of other network technologies such as WiFi or Bluetooth 4.0. WiFi is available on most smartphones, provides sufficient performance but is power hungry. Dynamic interface switching mechanisms can be incorporated to achieve the trade-off between energy and sensitivity. Bluetooth 4.0 provides ZigBee like performance of low power consumption and fast switching time. Although its availability is increasing, it is still not widely available yet.

Finally, in the case when all speakers rarely speaks and conversation last for a very short time (short cold conversation), no enough information can be collected for conversation clustering, and this contribute most of the errors in our clustering evaluation. To complement the current speaking-pattern-based clustering, other features such as facing direction, speech rhythm synchronicity and emotion synchronicity are all possible improvements in the future.

12. CONCLUSIONS

In this paper, we presented SocialWeaver, a sensing system running on smartphones to perform conversation clustering and build real-time conversation networks. SocialWeaver exploits collaboration among users to build proximity group, classify speaker, aggregate information and perform conversation group clustering. Our result shows that conversations clustering can achieve accuracy of between 71% to 92% accuracy and can derive the correct conversation clusters from both week-long and social event experiments. SocialWeaver provides a practical and effective platform for understanding human communication that has the potential for extracting real world social interactions and has many future applications.

13. ACKNOWLEDGEMENT

This research was carried out at the SeSaMe Centre. It is supported by the Singapore NRF under its IRC@SG Funding Initiative and administered by the IDMPO.

14. REFERENCES

[1] S. Baatz, M. Frank, C. Kuhl, P. Martini, and C. Scholz. Bluetooth scatternets: An enhanced adaptive scheduling scheme. In *INFOCOM*, 2002.

[2] S. Basu. *Conversational scene analysis.* PhD thesis, MIT, 2002.

[3] C. Bird, A. Gourley, P. Devanbu, M. Gertz, and A. Swaminathan. Mining email social networks. In *MSR*, 2006.

[4] A.T. Campbell, S.B. Eisenman, N.D. Lane, E. Miluzzo, R.A. Peterson, H. Lu, X. Zheng, M. Musolesi, K. Fodor, and G.S. Ahn. The rise of people-centric sensing. *Internet Computing*, 2008.

[5] T. Choudhury and A. Pentland. Sensing and modeling human networks using the sociometer. In *ISWC*, 2003.

[6] S.R. Corman and C.R. Scott. A synchronous digital signal processing method for detecting face-to-face organizational communication behavior. *Social Networks*, 1994.

[7] E. Cuervo, A. Balasubramanian, D. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl. Maui: making smartphones last longer with code offload. In *MobiSys*, 2010.

[8] S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *TASSP*, 1980.

[9] R. Honicky, E.A. Brewer, E. Paulos, and R. White. N-smarts: networked suite of mobile atmospheric real-time sensors. In *NSDR*, 2008.

[10] L.P. Kaelbling, M.L. Littman, and A.R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 1998.

[11] E. Koukoumidis, L.S. Peh, and M.R. Martonosi. Signalguru: leveraging mobile phones for collaborative traffic signal schedule advisory. In *MobiSys*, 2011.

[12] H. Kurniawati, D. Hsu, and W.S. Lee. Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In *Proc. Robotics: Science and Systems*, 2008.

[13] H. Kwak, C. Lee, H. Park, and S. Moon. What is twitter, a social network or a news media? In *WWW*, 2010.

[14] N.D. Lane, M. Mohammod, M. Lin, X. Yang, H. Lu, S. Ali, A. Doryab, E. Berke, T. Choudhury, and A.T. Campbell. Bewell: A smartphone application to monitor, model and promote wellbeing. In *PervasiveHealth2011*, 2011.

[15] Y. Lee, C. Min, C. Hwang, J. Lee, I. Hwang, Y. Ju, C. Yoo, M. Moon, U. Lee, and J. Song. Sociophone: Everyday face-to-face interaction monitoring platform using multi-phone sensor fusion. In *MobiSys*, 2013.

[16] Q. Li, J. Zheng, A. Tsai, and Q. Zhou. Robust endpoint detection and energy normalization for real-time speech and speaker recognition. *TSAP*, 2002.

[17] H. Lu, Bernheim A.B., B. Priyantha, A. Karlson, and J. Liu. Speakersense: Energy efficient unobtrusive speaker identification on mobile phones. *Pervasive Computing*, 2011.

[18] H. Lu, J. Yang, Z. Liu, N.D. Lane, T. Choudhury, and A.T. Campbell. The jigsaw continuous sensing engine for mobile phone applications. In *SenSys*, 2010.

[19] E. Miluzzo, C.T. Cornelius, A. Ramaswamy, T. Choudhury, Z. Liu, and A.T. Campbell. Darwin phones: the evolution of sensing and inference on mobile phones. In *MobiSys*, 2010.

[20] E. Miluzzo, N.D. Lane, S. Eisenman, and A. Campbell. Cenceme–injecting sensing presence into social networking applications. *Smart Sensing and Context*, 2007.

[21] H. Misra, S. Ikbal, H. Bourlard, and H. Hermansky. Spectral entropy based feature for robust asr. In *ICASSP*, 2004.

[22] P. Mohan, V.N. Padmanabhan, and R. Ramjee. Nericell: rich monitoring of road and traffic conditions using mobile smartphones. In *SenSys*, 2008.

[23] Rieks Akker op den, Daniel Gatica-Perez, and Dirk Heylen. Multi-modal analysis of small-group conversational dynamics. In *Multimodal Signal Processing*. 2012.

[24] J. Ramırez, J.C. Segura, C. Benıtez, A. De La Torre, and A. Rubio. Efficient voice activity detection algorithms using long-term speech information. *Speech Communication*, 2004.

[25] D.A. Reynolds. Speaker identification and verification using gaussian mixture speaker models. *Speech Communication*, 1995.

[26] D.A. Reynolds and R.C. Rose. Robust text-independent speaker identification using gaussian mixture speaker models. *TSAP*, 1995.

[27] P. Shockley-Zalabak. *Fundamentals of organizational communication.* 2011.

[28] J. Sohn, N.S. Kim, and W. Sung. A statistical model-based voice activity detection. *Signal Processing Letters*, 1999.

[29] G. Tan. *Self-organizing Bluetooth scatternets.* PhD thesis, MIT, 2002.

[30] D. Wyatt, T. Choudhury, J. Bilmes, and J.A. Kitts. Inferring colocation and conversation networks from privacy-sensitive audio with implications for computational social science. *TIST*, 2011.