Crowdsensing the Speaker Count in the Wild: Implications and Applications

Chenren Xu, Sugang Li, Yanyong Zhang *Rutgers University*
Emiliano Miluzzo, Yih-Farn Chen *AT&T Labs - Research*

***Abstract*—The Mobile Crowd Sensing (MCS) paradigm enables large-scale sensing opportunities at lower deployment costs than dedicated infrastructures by utilizing the large number of today's mobile devices. In the context of MCS, end users with sensing and computing devices can share and extract information of common interest. In this article, we examine Crowd++, a MCS application, which accurately estimates the number of people talking in a certain place through unsupervised machine learning analysis on audio segments captured by mobile devices. Such a technique can find application in many domains, such as crowd estimation, social sensing, and personal well-being assessment. In this article, we demonstrate the utility of this technique in the context of conference room usage estimation, social diary, and social engagement in a power efficient manner followed by a discussion on privacy and possible optimizations to Crowd++ software.**

I. Introduction

Mobile Crowd Sensing (MCS) [1], [2] is a sensing paradigm that complements traditional static-only deployments [3]. MCS leverages human mobility to achieve a larger sensing scale with lower infrastructure support, especially in places with frequent human activities. Broadly speaking, MCS can find application in both ambient sensing and social sensing, with ambient sensing mainly targeting the monitoring or tracking of physical phenomena such as pollution level, traffic congestion, and parking availability. In the social sensing sphere MCS applications can provide people with opportunities to connect through social networks, share personal data and form social interactions. In this article, we focus on the utility of MCS in the context of social sensing.

The most direct form of social interaction occurs through the spoken language and conversations. Given its rich context, for decades scientists have proposed diverse methodologies to analyze the audio recorded during people's conversations to characterize this particular social interaction through various attributes such as speech content (what), speaker identification (who) and emotion detection (how). Though these are excellent drivers for many social sensing applications, there are two potential limitations with the current approaches. First, they need pre-labeled training data to train classifiers. Second, while the analysis of their sensor data can be meaningful for individuals, drawing statistical conclusions for crowds is very challenging.

We note that one of the most important contextual attributes of a conversation, namely, speaker count, has been largely overlooked and can break new ground in MCS applications. Speaker count specifies the number of people that participate in a conversation, which is one of the primary metrics to evaluate a social setting: how crowded is a restaurant, how interactive is a lecture or a meeting, or how socially active a person is [4], [5].

Given that mobile devices are becoming increasingly powerful and ubiquitous, it is natural to envision new social monitoring architectures, with the collection of these devices being the only sensing and computing platform. In pursuit of these goals, we design a system called Crowd++, where we exploit the audio from the smartphones microphone to draw the social fingerprints of a place, an event, or a person. Although mobile audio inference has been previously used to characterize places and events by picking up different sound cues in the environment [6], [7], we tackle a complete new angle. We infer the number of people in a conversation – but not their identity[1] – as well as their interactions from the analysis of the voices contained in audio snippets captured by the smartphones, *without any prior knowledge of the speakers and their speech characteristics* [8].

One question that comes to mind is: Why do we need a solution like Crowd++ to infer the number of people in a place? Wouldn't it be enough to simply count the number of WiFi devices associated with an access point, piggyback to a bluetooth scan result, measure co-location, use computer vision techniques to analyze the number of people in video images, or even use active methods that require the transmission and analysis of audio tones? The answers to these questions are quite straightforward: none of these techniques in isolation is the solution to the problem. In order to read the association table of an access point there is a need to have access to the WiFi infrastructure, which is often not allowed. Even if possible, a person with several WiFi devices may generate false positives. A count based on the result of a bluetooth discovery [9] is error-prone because of the likelihood of reaching out to distant devices. RF-based device-free localization techniques [10] require the support of an infrastructure of several radio devices. Acoustic-based counting engines as in [11] are error-prone because of surrounding noise and audio sensitivity to clothes. Counting people through computer vision techniques [12] requires customized infrastructure, suffers from privacy concerns, and is limited by lighting condition. Crowd++ inference is instead based on a much more localized event – speech – that can significantly scope the count inference to specific geographic regions. It's also passive, since no active sounds by the devices need to be

[1] Except for the social diary use case, where we want to measure how active a particular person is in his or her daily social interactions.

played.

We also acknowledge that in other places, such as subway stations or movie theaters, silence or loud noise can dominate, making it difficult for Crowd++ to properly operate. We, however, note that Crowd++ should not be deemed as a replacement of any of the existing approaches. Rather, it should be seen as a complementary solution that can be useful to boost the crowd count accuracy by working in concert with different techniques. Prior information about a certain place – such as the average number of people attending the place – combined with the properties of statistical sub-sampling can also be used to boost the final count accuracy.

We have implemented Crowd++ on four Android smartphones and two tablet computers and collected over 1200 minutes of audio over the course of three months from 120 different people. The audio is recorded by Crowd++ in a range of different environments, from quiet ones – home and office – to noisy places like restaurants, malls, and public squares. We show that the average difference between the actual number of speakers and the inferred count with Crowd++ is slightly over 1 for quiet environments, and no larger than 2 in very noisy outdoor environments [8]. While Crowd++ may be deemed only as an initial step, we show that faithful people count estimates in conversations can nevertheless be achieved with sufficient accuracy. We conjecture that this accuracy is adequate and meaningful for many applications – such as social sensing applications, crowd monitoring and social hotspots characterization just to name a few – where exact count estimation is not necessarily a requirement.

The code release of the Crowd++ application can be found here [13].

II. Unsupervised Speaker Counting

The knowledge of a person's social activity level, or a social setting's popularity, is important for many applications. These two scenarios appear very different, but they can be both characterized by the number of active speakers in the conversation. To estimate the number of active speakers in a group of people, we decompose this process into three steps: (1) *speech detection*, (2) *feature extraction*, and (3) *counting*. In the speech detection phase, we discard the silence periods and background noise, and only keep the human voice segments. In the feature extraction phase, we then extract the feature which can uniquely characterize each speaker from the speech data. In the counting phase, we use a distance function to characterize the dissimilarity if the two speech segments come from different speakers, and then apply an unsupervised learning technique that, operating on the feature vectors with the support of the distance model, finally determines the speaker count. We refer to this application as *Crowd++* and we show the overview of the Crowd++ pipelined approach in Figure 1.

A. Speech Features and Distance Metrics

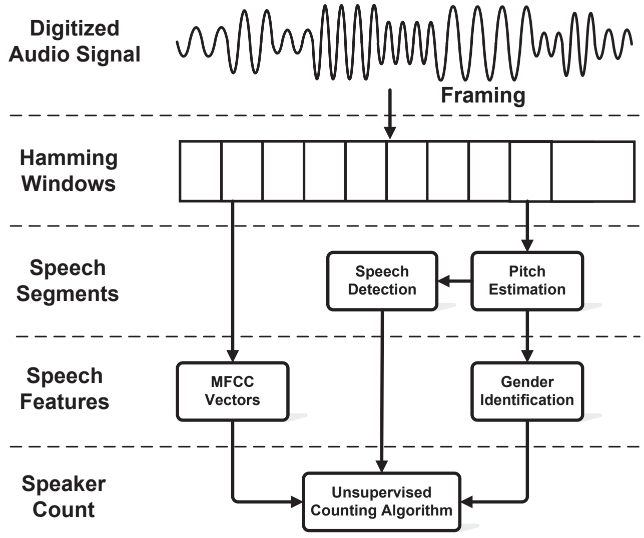


Fig. 1. The sequence of operations in Crowd++.

Crowd++ learning algorithm is based on a feature vector composed by pitch and mfcc features, as largely used in speech processing algorithms. A distance model to quantify the dissimilarity between feature vectors, hence between speakers is employed. We first introduce a few terms that are essential to our counting algorithm.

Pitch: In speech, pitch is the relative highness or lowness of a tone perceived by the ear, which depends on the number of vibrations per second produced by the vocal cords. We have two general observations about pitch. First, human pitch is distinctively different from that of other creatures, or sounds from objects. Second, male's pitch is statistically lower than female's. As a result, given an audio segment, we can use its pitch value to tell whether there is any human speech in the segment, and further, to tell the gender of the speaker.

MFCC: MFCC (Mel Frequency Cepstrum Coefficients) is a representation of the short-term power spectrum of a sound, based on a linear cosine transformation of a log power spectrum on a nonlinear mel scale of frequency, which approximates the sensitivity of the human ear. Thus, it can efficiently characterize each speaker. Usually, the vector containing the 2nd-13th coefficients is good enough for human voice characterization.

Feature Distance: After obtaining the speech feature vectors, we need a distance model to quantify the dissimilarity between the voice-feature vectors. We select the cosine similarity distance, which is essentially the angle between two vectors in the geometric hyperplane. With this distance model, we can get a general idea of the distance distribution of speech vectors from the same speaker and speech vectors from different speakers. Such distributions are key to our speaker counting process.

B. Speaker Counting Process

Here we discuss the speaker counting algorithm. The first step is to segment the audio recording into n equal length audio segments. For each segment, we compute its pitch and MFCC vectors. After detecting whether there is human speech in the segment based upon its pitch value, we filter out noises and only keep those speech segments.

The main idea for speaker counting is to distinguish different speakers from the audio segments. Thus, the most critical part is speaker distinction, i.e., to identify if the two audio segments come from the same speaker or not. Given two speech segments, we tell whether they belong to the same speaker as follows:

- *Same Speaker:* If the MFCC distance is less than θ_s and the pitch infers the same gender.
- *Different Speakers:* If the MFCC distance is larger than θ_d or the pitch infers different genders.
- *Uncertainty:* otherwise.

The thresholds, θ_s and θ_d, are empirically determined in the calibration phase before we conduct the evaluation. We note that the optimal threshold values may vary across different phone models because of different sensitivity levels of mobile device microphones. The choice of these two thresholds is driven by the desire to be conservative in the discovery of new speakers while minimizing the number of false positives. In our implementation, we first conducted a calibration phase by collect monologue speech from 10 participants (5 males and 5 females) from different countries with different accents with difference phone models. Next, we segment the their speech into smaller segments and compute the MFCC distance values from the same speaker and different speakers and denote them as $\mathcal{D}_s$ and $\mathcal{D}_d$ respectively. Finally, θ_s and θ_d are chosen as the median value from $\mathcal{D}_s$ and $\mathcal{D}_d$. During this process, each phone model collects its audio data and determines its own optimal values for θ_s and θ_d. More detailed information can be found in [13].

This speaker distinction function is the core of our counting algorithm, which consists of two steps.

Round One: Forward Clustering. Here we aggregate neighboring segments that produce similar features. More specifically, we merge the neighboring segments if the speaker distinction function tells that they come from the speaker. This clustering algorithm scans each segment once resulting in a linear time complexity. The rationale behind forward clustering is that there usually exists temporal correlation in speech – the likelihood of contiguous segments containing the same voice is high when the segments are short enough. After running the forward clustering algorithm, we have fewer and longer segments. We also note that longer segments have better performance in distinguishing different speakers and further boosting counting accuracy.

Round Two: Speaker Counting. At this stage, we first label the first speech segment as the first speaker. Next, we keep scanning the following segments by applying our speaker distinction function. If the new speech segment comes from one or more of the identified voices, we merge it with the most similar one. On the contrary, we count a new speaker if and only if it is different from all the identified voices. We discard the speech segments that are not distinguishable enough, such as the speech with partial silence, overlap, or unvoiced data. For more details about the Crowd++ learning algorithm please refer to [8].

Crowd++ is designed to operate in a fully distributed manner, with the inference being performed by each smartphone individually. However, a more cooperative approach as in [14] involving multiple devices at the same time in the inference process, could also be adopted to further boost the inference accuracy.

As we discuss in the following section, Crowd++ can be harnessed in different MCS application scenarios, some of them being, for instance, the assessment of people's social activity level or popularity of a certain place. In the rest of this section, we discuss three possible Crowd++ applications.

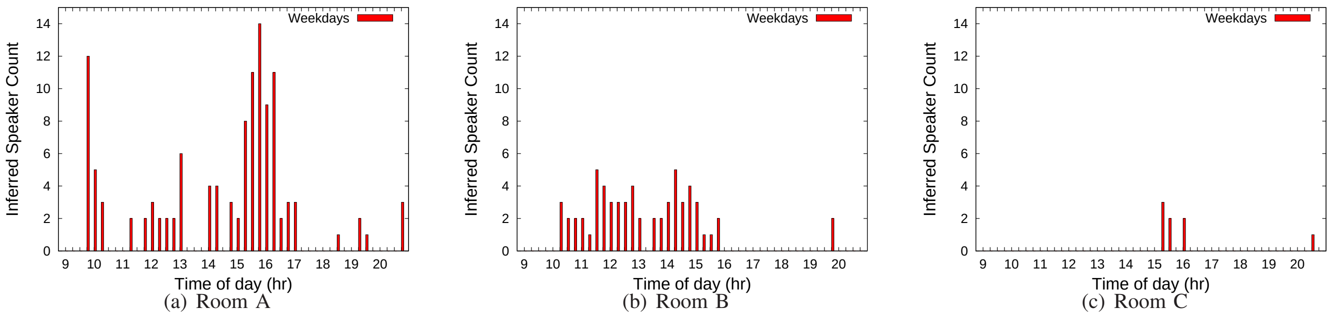
III. Crowd++ Applications in Mobile Crowd Sensing

A. Crowd Estimation and Social Hotspots Discovery

We assume that people usually engage in conversations in social public spaces such as restaurants, pubs, student center, or meeting rooms. Therefore, we can use Crowd++ to estimate the number of people talking, and use this information to assess the "crowded-ness" of these places. As a proof-of-concept, we have deployed three smartphones running Crowd++, each in a different conference room at WINLAB, Rutgers University, to track the occupancy of each room. We have written a *HotRoom* smartphone app that samples the room every 15 minutes by recording the sound in the room for 5 minutes. We continuously ran this app in all three conference rooms for two weeks.

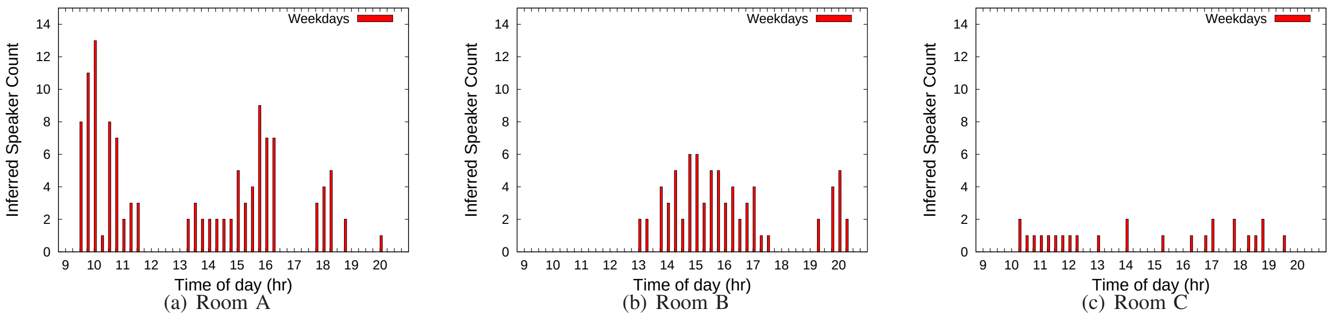
The three conference rooms at WINLAB are shown in Figure 4. Two of them are in the main area (Room B and C), while the third one (Room A) is located in the left wing of the building that is connected to the main area through a long hallway. Room C is much larger than Room B, with the former holding 30 seats while the latter only 6 seats. The third one, Room A, can hold 8 seats. Usually Room A and B are more popular than Room C because a large portion of the meetings are often with only a small number of participants. This is further confirmed by the occupancy data of the three conference rooms that are observed Monday through Friday, as shown in Figure 2 and 3, where we sum up the inferred speaker count across different workdays from 9 am to 9 pm for two weeks. Among the three conference rooms, A is the most popular one due to its more private location. For Room A, we observe two regular patterns: around 10 am, which is the regular meeting time for a WINLAB spinoff startup, and around 3:30 pm which is when most of the faculty meet their students. Room B is mostly occupied in the afternoon. Among the three conference rooms, room C is the least popular because it is a much larger space, usually reserved only for big events such as Ph.D. defense. Nonetheless, in many cases, only one person is inferred even in Room C. This can be an underestimation of the ground truth because usually the presenter talks dominantly during his/her defense.

We also show the heat map of conference room usage at 3:30 pm on Tuesday of the first week in Figure 4. This pictorial representation is quite useful and helps people realize the status of a room when looking for empty spaces. Without



(a) Room A (b) Room B (c) Room C

Fig. 2. Room occupancy at WINLAB during the first week of the Crowd++ deployment.



(a) Room A (b) Room B (c) Room C

Fig. 3. Room occupancy at WINLAB during the second week of the Crowd++ deployment.

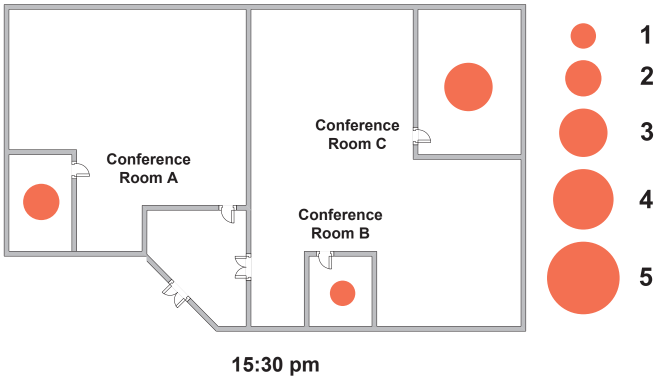


Fig. 4. The hotmap of WINLAB conference usage at 15:30 pm on 2013.11.12.

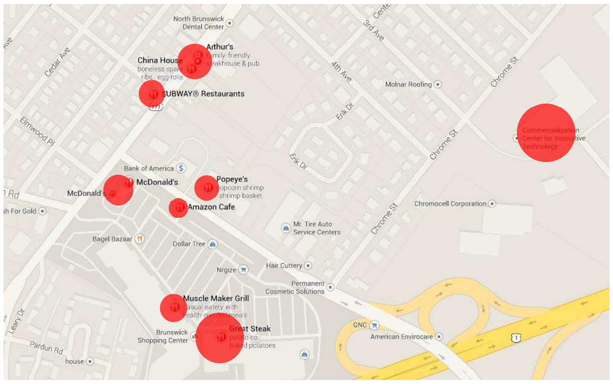


Fig. 5. Our envisioned hotspot map in Rutgers when speaker count results are obtained from large-scale analysis of crowd audio.

Crowd++ there would be a need to deploy costly and static sensing infrastructures.

With Crowd++ running at large-scale we could provide town-wide heat maps of the most popular social places, as shown in Figure 5. This information could be used, for example, to augment local search results showing the most popular bars, restaurants, stores, and venues where people gather in large groups.

B. Personal Social Diary.

Doctors analyze their patients' social patterns to predict depression or social isolation and take early actions. Rather than using ad-hoc hardware [4], which could potentially perturb the quality of the measurements, Crowd++ can be installed on the smartphones of people potentially affected by depression to monitor their social engagement in a more scalable, and less invasive manner. Crowd++ would operate transparently as these people go about their daily lives. To this goal, Crowd++ could also be used to build a person's social diary, inferring how many people the person talks to and at what time, which is a useful tool to seniors, Alzheimer patients, and people with clinic depression.

As a proof-of-concept, we have recruited three participants, a teacher, a student, and a company employee, to use our SocialDiary supported by Crowd++ app to record their conversations for a week. This app records audio every 2 hours for 8 minutes. We show the log for the student on a weekday and a weekend in Figure 6. From the social diary, we observe that the student has very different social patterns on workdays than weekends: on a typical weekday, she gets up early, attends a

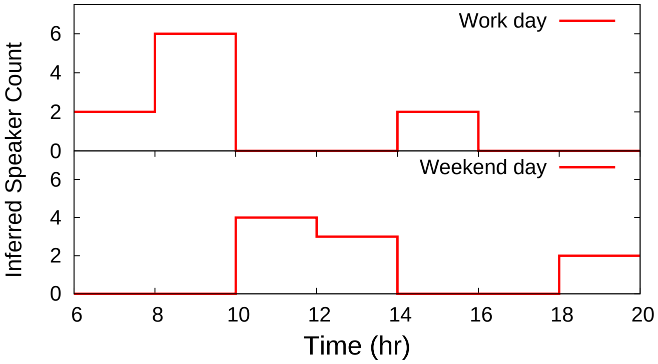


Fig. 6. The social diary of a participant shows that she has different social patterns on workdays and weekends.

class with active discussion, attends work meetings with his advisor in the afternoon, and has lunch and dinner mostly alone. On weekends, her pattern is flipped: she gets up very late, has a long brunch with his friends, and dinner with a family member.

C. Event Engagement Estimation.

Crowd++ can be used to estimate how engaging an event is, i.e., the level of interaction between speakers. For example, as a student, you may want to take the class where the teacher promotes interactions with students during lectures. This feature could be captured by the number of people who talk during a lecture. Such a measurement could allow parents to be aware of their kids' participation in the classroom as well. Or, for those attending large public meetings with potentially multiple working groups. In this scenario, an attendee may want to join the group with the most active discussions. In both scenarios, Crowd++ can be used to capture the level of engagement in the event with interesting social implications.

As a preliminary experiment, we have recorded 2 lectures and 2 recitation sessions on a university campus and 4 seminars from an industrial working space. Each recording lasts 60 minutes. We segment each audio into six 10-minute segments and estimate the speaker count in each segment, as well as the total speaker count for the whole period. We show how the speaker count varies with time in Figure 7. We observe that the recitation involves less interaction of all – the instructor spends most of the time showing how to solve the homework problems on the blackboard. The regular class has a steady interaction level throughout the duration, while the seminar presents more questions at the beginning.

A more comprehensive set of results about Crowd++ performance in different scenarios, including noisy environments can be found in [8].

IV. Energy Consumption

In order to operate correctly and accurately, ideally Crowd++ should run continuously on a mobile device. Continuous audio recording and processing, however, would severely impact the battery performance, and necessary precautions need to be taken. We adopt a duty-cycling approach to sensing and inference. In our implementation, Crowd++ records audio for 5 minutes every 15 minutes. The Crowd++ count inference is performed upon every recording. We chose the HTC EVO 4g, Samsung Galaxy S2, S4, Google Nexus 4 and Motorola Moto X as five different devices to test the battery duration with these parameters. Our measurements in Figure 8 show that the battery can last more than 24 hours using the latest smartphone devices. All the measurements are collected with the WiFi service running in background on the phone and include both the audio recording and processing. These battery durations are compatible with the normal usage of a phone, which is typically recharged at night. We note that these battery durations are achieved with a fixed duty-cycle, providing a lower bound on the battery lifetime. Energy boost could be obtained, for example, by running Crowd++ only in public places. This policy would serve two purposes: longer battery life and privacy in private spaces.

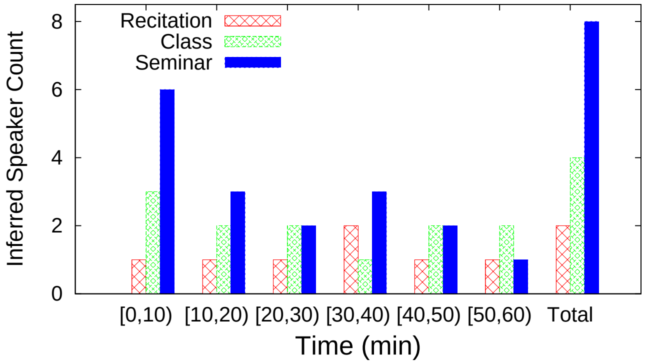


Fig. 7. Seminars and classroom lectures have different interaction patterns over the time.

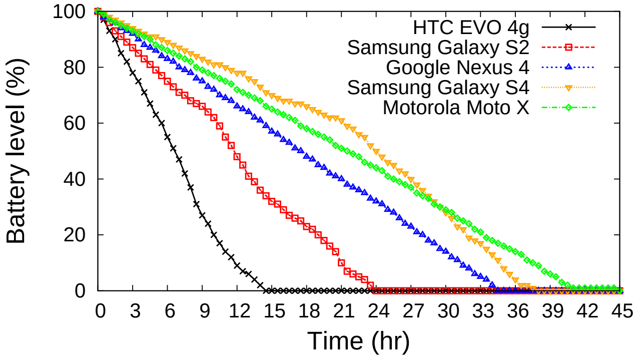


Fig. 8. A duty-cycle of 15 mins guarantees a one day battery life for modern smartphones such as Samsung Galaxy S2, S4, Google Nexus 4 and Motorola Moto X.

V. Discussion

In this section, we discuss the energy optimization, privacy concern, and feasibility of Crowd++.

A. How can we further conserve energy?

Further conserving energy consumption (beyond fixed duty cycling) presents unique challenges. Inferring speaker count is data intensive for both recording and processing. Unlike speech recognition and speaker identification in which the required speech data are relatively short e.g., around 10 seconds, speaker count can only be extracted from a conversation if we have at least a few minutes of audio data in order to capture everyone's voice. To conserve energy, the ideal policy is to turn on the microphone when and only when the conversation takes place, which is hard to implement unless a low-power secondary mic, like in the Motorola Moto X is available. Instead, we can explore the following approaches.

Heterogenous Processing: One way to achieve energy efficiency is through the addition of a hardware dongle that separates speech detection and audio processing. For example, SpeakerSense [15] uses an external MSP430 based hardware dongle to detect speech in a low power mode and wake up the smartphone "just in time" for audio processing. Another way is to assign the speech processing task to a dedicated core to achieve energy efficiency. For example, Moto X is equipped with such a core in its CPU architecture so that all the speech-related processing can be done on this core that is optimized for such tasks.

Context-based Opportunistic Sampling: We can also explore the possibility of predicting when a conversation takes place and turn on the speaker counting service accordingly. Next we explain how this may work using a method driven by a personal social diary. The main idea is that we can learn a person's conversation patterns with respect to various personal/environmental contexts.

- *Proximity Detection.* Given that it is safe to assume that people always carry a smartphone, we can detect phone proximity through short-range radio analysis techniques to determine whether two phones are close to each other, which is an indicator that the people may engage in a conversation. Proximity may be then used as a soft hint for Crowd++ to start audio sensing.
- *Lightweight Sampling.* A more efficient sensing and processing duty-cycling technique could be adopted. For example, the duty-cycle could follow a backoff policy, where the microphone samples for a small-duration just to detect voices. If no voice is detected then an exponential backoff could be applied to the sensing sleeping time. The backoff is reset when a voice is detected.
- *Learning-Based Prediction.* We can collect a person's conversation log for a period of time, and learn her conversation pattern. For example, we may be able to correlate the likelihood of the user to talk to others with time of the day, calendar events, or certain phone usage patterns. Crowd++ could again use this as a hint that could drive smarter duty-cycling policies.

B. Privacy

It is quite natural to raise privacy concerns when doing audio analysis. These concerns become more serious when the audio is captured with a smartphone, which is often in private spaces, e.g., home. With this in mind, we take specific steps to make sure that users' privacy is preserved.

Speakers' identity is never revealed. Crowd++ isn't able to associate a voice fingerprint to a specific person and it's designed to only infer the number of different speakers in an anonymized manner. Crowd++ could potentially identify only the phone's owner if the algorithm was actively trained to recognize the owner's voice. Identification of the owner may be optionally added to either improve the speaker count accuracy or used in personal social diary applications.

The audio analysis is always performed locally on devices in order to avoid sensitive data leaks. The audio is deleted right after the audio features computation. Should communication with backend be needed, the servers should be trusted and off-the-shelf encryption methods for the communications should be put in place. Only features extracted from the audio, rather than the raw audio itself, should be sent to the server.

To guarantee the user's privacy when the data is sent to a backend server and to prevent attacks that exploit the audio features to reconstruct the original audio, measures such as the ones proposed by Liu et al. [16] should be put in place. In this work, it is shown how to manipulate the MFCC features to a point that they are still effective for a machine learning algorithm to infer events while, however, obfuscating the underlying content of the raw audio.

Finally, by giving users the ability to configure the application's settings, Crowd++ should be allowed to work only in specific locations – say, in public places. Through geo-fencing technologies, the application could be automatically activated and deactivated as directed by the user's pre-selected policies: e.g., activate it in the office and in restaurants but not at home.

C. Feasibility of Crowd++

We acknowledge that the performance of Crowd++ is affected by the smartphone location and surrounding audio context. For example, the audio quality degrades when the phones are in pockets or backpacks. Nevertheless, as the popularity of sensing-equipped mobile devices and wearables, e.g., smart glasses, watches, bracelets, and rings (Figure 9) increases, more device can participate in the inference process offsetting the errors of any individual inference. Moreover, in a context of pervasive and ubiquitous continuous sensing enabled by all these devices, we expect a steady growth of services and applications designed to infer people context and behavior to augment an add new dimensions to existing social, medical, and utility applications.

VI. Conclusion

In this article, we presented an example of a MCS instance called Crowd++, a scalable, privacy-aware, and energy-efficient speaker-count application based on audio analysis from mobile devices' microphones. In contrast to more complex and less scalable counting techniques, Crowd++ is a lightweight approach that can support many different application scenarios: from social sensing – to determine social hotspots – to personal wellbeing assessment and social diary and social engagement, place characterization, and more accurate localization techniques.

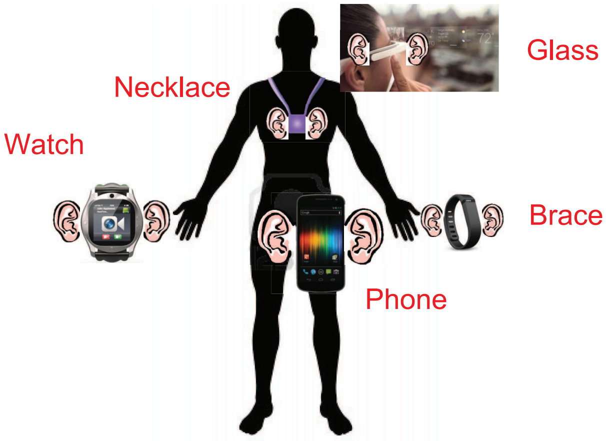


Fig. 9. Feasibility of audio inference.

References

[1] 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," *IEEE Internet Computing*, 2008.

[2] R. K. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: Current state and future challenges," *IEEE Communications Magazine*, 2011.

[3] C. Ozturk, Y. Zhang, W. Trappe, and M. Ott, "Source-location privacy for networks of energy-constrained sensors," in *IEEE WSTFEUS*, 2004.

[4] M. Rabbi, S. Ali, T. Choudhury, and E. Berke, "Passive and in-situ assessment of mental and physical well-being using mobile sensors," in *ACM UbiComp*, 2012.

[5] A. Matic, V. Osmani, and O. Mayora, "Automatic sensing of speech activity and correlation with mood changes," *Pervasive and Mobile Sensing and Computing for Healthcare*, 2012.

[6] H. Lu, W. Pan, N. D. Lane, T. Choudhury, and A. T. Campbell, "Soundsense: scalable sound sensing for people-centric applications on mobile phones," in *ACM MobiSys*, 2009.

[7] M. Azizyan, I. Constandache, and R. Roy Choudhury, "Surroundsense: mobile phone localization via ambience fingerprinting," in *ACM MobiCom*, 2009.

[8] C. Xu, S. Li, G. Liu, Y. Zhang, E. Miluzzo, Y.-F. Chen, J. Li, and B. Firner, "Crowd++: Unsupervised speaker count with smartphones," in *ACM UbiComp*, 2013.

[9] J. Weppner and P. Lukowicz, "Collaborative crowd density estimation with mobile phones," in *ACM PhoneSense*, 2011.

[10] C. Xu, B. Firner, R. S. Moore, Y. Zhang, W. Trappe, R. Howard, F. Zhang, and N. An, "Scpl: indoor device-free multi-subject counting and localization using radio signal strength," in *ACM/IEEE IPSN*, 2013.

[11] P. G. Kannan, S. P. Venkatagiri, M. C. Chan, A. L. Ananda, and L.-S. Peh, "Low cost crowd counting using audio tones," in *ACM SenSys*, 2012.

[12] A. B. Chan, Z.-S. Liang, and N. Vasconcelos, "Privacy preserving crowd monitoring: Counting people without people models or tracking," in *IEEE CVPR*, 2008.

[13] "The crowd++ android app source code repository," https://github.com/lendlice/crowdpp.

[14] 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 *ACM MobiSys*, 2010.

[15] H. Lu, A. B. Brush, B. Priyantha, A. K. Karlson, and J. Liu, "Speakersense: energy efficient unobtrusive speaker identification on mobile phones," in *Pervasive*, 2011.

[16] B. Liu, Y. Jiang, F. Sha, and R. Govindan, "Cloud-enabled privacy-preserving collaborative learning for mobile sensing," in *ACM SenSys*, 2012.

Chenren Xu (chenren@cs.cmu.edu) is a Postdoctoral Research Fellow in the School of Computer Science at Carnegie Mellon University. His current research interests include energy-efficient computing, mobile and ubiquitous sensing, context-awareness, wearable computing, wireless networking, machine learning and human computer interaction. He holds a Ph.D. in Electrical and Computer Engineering and a M.S. in Applied Mathematical Statistics from Rutgers University, and a B.E. (with highest honors) in Automation from Shanghai University.

Sugang Li (sugangli@winlab.rutgers.edu) is a Ph.D. candidate in the Department of Electrical and Computer Engineering and a member of WINLAB at Rutgers University. His research interest includes mobile sensing, indoor localization, Internet-of-Things (IoT) of Future Internet Architecture.

Yanyong Zhang (yyzhang@winlab.rutgers.edu) is currently an Associate Professor in the Electrical and Computer Engineering Department at Rutgers University. She is also a member of the Wireless Information Networks Laboratory (Winlab). Her current research interests are in sensor networks and pervasive computing. Her research is mainly funded by the National Science Foundation, including an NSF CAREER award.

Emiliano Miluzzo (miluzzo@research.att.com) is an experimental researcher working at the intersection of mobile systems and applied machine learning in the Mobile and Pervasive Systems Research group at AT&T Labs Research. His research interests include mobile, pervasive, distributed computing, mobile sensing systems, and big data analysis. He holds a Ph.D. in Computer Science from Dartmouth College, and a M.Sc. and B.Sc. in Electrical Engineering from University of Rome "La Sapienza", Italy.

Yih-Farn Chen (chen@research.att.com) is a researcher in the Mobile and Pervasive Systems Research group at AT&T Labs Research. His current research interests include cloud computing, mobile computing, distributed systems, World Wide Web, and IPTV. He holds a Ph.D. in Computer Science from University of California at Berkeley, an M.S. in Computer Science from University of Wisconsin, Madison, and a B.S. in Electrical Engineering from National Taiwan University. Robin is an ACM Distinguished Scientist and a Vice Chair of the International World Wide Web Conferences Steering Committee (IW3C2). He also serves on the editorial board of IEEE Internet Computing.