SymDetector: Detecting Sound-Related Respiratory Symptoms Using Smartphones

Xiao Sun, Zongqing Lu, Wenjie Hu, Guohong Cao
Department of Computer Science and Engineering
The Pennsylvania State University, University Park, PA, 16802, USA
{xxs118, zongqing, wwh5068, gcao}@cse.psu.edu

ABSTRACT

This paper proposes *SymDetector*, a smartphone based application to unobtrusively detect the sound-related respiratory symptoms occurred in a user's daily life, including sneeze, cough, sniffle and throat clearing. SymDetector uses the built-in microphone on the smartphone to continuously monitor a user's acoustic data and uses multi-level processes to detect and classify the respiratory symptoms. Several practical issues are considered in developing SymDetector, such as users' privacy concerns about their acoustic data, resource constraints of the smartphone and different contexts of the smartphone. We have implemented SymDetector on Galaxy S3 and evaluated its performance in real experiments involving 16 users and 204 days. The experimental results show that SymDetector can detect these four types of respiratory symptoms with high accuracy under various conditions.

ACM Classification Keywords

C.3. Special-purpose and Application-based Systems: Real-time and embedded systems

Author Keywords

Respiratory symptom detection; microphone; feature extraction; smartphone.

INTRODUCTION

Respiratory symptoms are related to illnesses, infections or allergies. Among such symptoms, sound-related respiratory symptoms, such as sneeze, cough, sniffle and throat clearing, are commonly observed and useful in health-related research. For example, by collecting self-reported flu symptoms including aforementioned ones from registered users every week, a nationwide flu map is built in [6] to illustrate how flu spreads. In [7], self-reported symptom data including cough and dry throat is collected to study the relationship between student health and indoor air quality in schools. However, self-reporting, which has been commonly used in the current research to collect respiratory symptom data, has been shown to be inefficient and inaccurate in [30, 15].

UbiComp '15, September 7-11, 2015, Osaka, Japan.

http://dx.doi.org/10.1145/2750858.2805826

To deal with this issue, in this paper, we present a practical system *SymDetector* to help researchers collect accurate sound-related respiratory symptom data from users by using off-the-shelf smartphones. SymDetector leverages the built-in microphone sensor to sense the phone's acoustic context and detect the user's acoustic events which are related to respiratory symptoms, including sneeze, cough, sniffle and throat clearing. SymDetector can work in an unobtrusive way to collect users' symptoms for a long period and the detection results can be provided to help medical research.

For certain types of symptoms such as sneeze and cough, there has been some research on how to detect them [25, 13, 32, 24, 15]. However, approaches proposed in [25] and [13] are not practical since users have to wear specialized sensors (piezoelectric sensor in [25] and accelerometer in [13]) on their chests to detect coughs. Audio based schemes for sneeze and cough detection have been proposed in [32] and [24], but they need to record users' audio data all day long and cannot work in real time. Larson *et al.* [15] proposed a real-time cough detection system by implementing machine learning techniques on smartphones. However, to sample a user's acoustic data, the phone has to be in a specific position (around the user's neck), and their system consumes lots of power (shown in *Performance Evaluations*).

Different from the aforementioned works, SymDetector detects four types of respiratory symptoms (i.e., sneeze, cough, sniffle and throat clearing) and considers several practical issues, such as users' privacy concerns about their acoustic data, resource constraints of the smartphone and different contexts of the smartphone. SymDetector consists of four components. *Audio Sampler* reads audio samples from the microphone and segments them as frames and windows. The windows which may potentially contain respiratory symptoms are sifted out by *Symptom Detector* and fed to *Symptom Classifier*, where acoustic features are extracted and multi-level classifiers are used to classify the respiratory symptoms. The detection results are then recorded in *Symptom Recorder*. SymDetector only buffers a window for a short time and the buffered window is discarded after being processed. All the acoustic data is processed locally and no raw data will be recorded on the phone permanently. SymDetector is designed to be lightweight and robust, so that it can work on a smartphone for a long time and detect respiratory symptoms under various contexts. We have implemented SymDetector on the Andorid based phone Galaxy S3 and evaluated its performance in real experiments involving 16 users and 204 days. The results

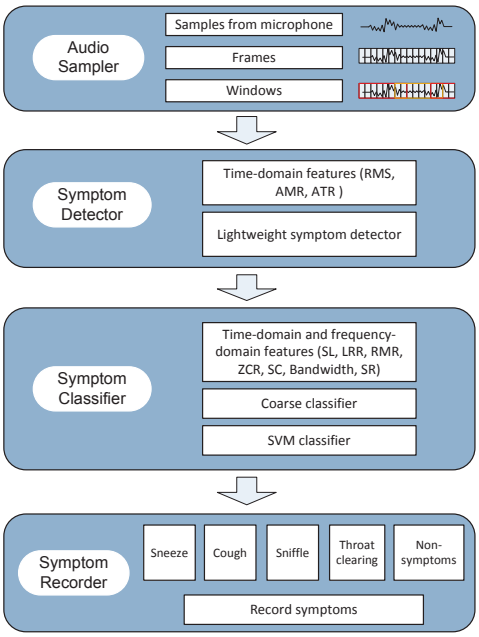


Figure 1: The architecture of SymDetector.

show that our system can detect respiratory symptoms with high accuracy under various conditions.

The rest of this paper is organized as follows. In the next section, we discuss the design considerations in developing SymDetector. Then, we present the system architecture and illustrate the design details of each component in SymDetector. The implementation of SymDetector on Android based phone and the evaluation of its performance in real experiments are demonstrated in the following sections. Then the implication is discussed and the related work is reviewed before concluding this paper.

DESIGN CONSIDERATIONS

SymDetector is designed to unobtrusively detect and record a user's respiratory symptoms occurred in his/her daily life using a smartphone. To achieve this goal, the following design issues should be considered.

First, users' privacy should be protected when SymDetector is working. Since SymDetector needs to monitor users' acoustic data, which may contain large quantities of personal and private information (e.g., conversations or background noise that may expose users' locations or activities), users will be concerned about how their acoustic data is sampled and used in the application. To protect users' privacy, no raw acoustic data is permanently recorded in SymDetector. SymDetector only stores a short period of samples temporarily to detect whether any respiratory symptom exists or not, and then these samples will be discarded. To prevent users' acoustic data from being disclosed, all the samples are processed locally and no raw data is transferred to remote servers. Eventually, only the detection results (e.g., the occurrence time and the type of each symptom) are kept and users can track them locally or upload them to some trusted servers.

Second, SymDetector must be lightweight. Since it is hard to know when a respiratory symptom will occur beforehand, the microphone must keep sensing users' acoustic data, which requires SymDetector to be able to process a large amount of raw acoustic data in real time. To preserve users' privacy [18, 17], all the sampled data must be processed locally on the smartphone, which has limited resources. Thus, SymDetector should be lightweight (i.e., it should consume less CPU and power). Although there are some existing sneeze [32, 33] and cough [15, 24, 23] detection schemes based on acoustic signal processing, they are not lightweight and cannot be directly applied in SymDetector.

Third, the positions of the smartphone with respect to the user (i.e., the context of the phone) should be considered to make SymDetector work unobtrusively and robustly. Due to users' different usage patterns, phones may work in various contexts. For example, some users prefer to put their phones on the desk when they work or study, while others prefer to put their phones in pockets or backpacks when phones are not used. Even for the same user, the context of his/her phone may change within a day. For example, a user may put his phone in the pocket when he works, but take it out of pocket when he wants to use the phone (e.g., sending texts, checking emails, playing games). According to the laws of acoustic wave propagation, the acoustic samples recorded by a phone will be affected by the context of the phone. Therefore, SymDetector should be able to detect respiratory symptoms in different contexts.

SYSTEM DESIGN

In this section, we present the design of SymDetector considering the above issues. As shown in Figure 1, SymDetector consists of four components. *Audio Sampler* is used to read acoustic samples from the microphone and segment them as frames and windows for further analysis. All windows of samples are processed by *Symptom Detector*. It sifts out the windows which may potentially contain respiratory symptoms and passes them to *Symptom Classifier*, where multi-level classifiers are used to classify each respiratory symptom. The detection results are recorded in *Symptom Recorder*. The design details of each component are described as follows.

Audio Sampler

The audio signals can be sampled at different sampling rates. For example, the microphone on Samsung's smartphone Galaxy S3 can work at 8 KHz, 11.025 KHz, 16 KHz, 22.05 KHz and 44.1 KHz. As the same in [10], the sampling rate in SymDetector is set to 16 KHz. Audio signals are continuously sampled from the microphone and each sample is represented by a 16-bit binary value.

The sampled audio stream is then segmented into non-overlapped frames of 50 ms (i.e., 800 samples) for feature extraction. As can be seen from Figure 2, which depicts the distribution of the symptom length based on our preliminary dataset, since respiratory symptoms may last for hundreds of milliseconds and cover several frames, it is difficult to determine whether a symptom occurs or not and when it occurs merely based on features extracted from one single frame.

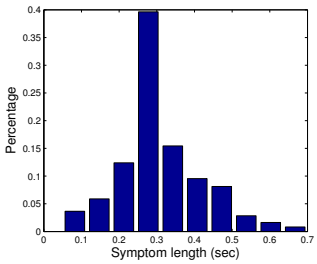


Figure 2: The distribution of symptom length.

Therefore, several continuous frames are grouped together as a window and each window is fed to the Symptom Detector as a unit for processing. The window in SymDetector is set to 4 seconds (i.e., 80 frames) so that it is long enough to cover the entire symptom and short enough to protect users' privacy. Since the windows are segmented based on the timeline, it is possible that a symptom may start at somewhere in the rear part of a window and end at the front part of the next window. In this case, though the window is larger than the event length, the entire event is not covered by any single window, which makes it difficult to extract window-level features correctly. In order to cover this window-crossing event in one window, the windows are built to overlap each other. As observed in Figure 2, even the longest symptom does not last more than 1 second. Thus in SymDetector, the overlap between windows is set to 1 second. Once a window of 64K samples (16K from the previous window and 48K newly sampled from microphone) is obtained, it is fed to the Symptom Detector.

Symptom Detector

Symptom Detector is designed as a filter to sift out the windows which potentially contain respiratory symptoms from the audio stream. The aim of the Symptom Detector is to use a lightweight scheme to filter out as many non-symptom windows as possible, leaving only a small number of windows to Symptom Classifier, where multi-level classifiers are used to classify them as sneeze, cough, sniffle, throat clearing or non-symptoms in a more precise way. Since in indoor environment, the predominant non-symptom windows are those containing either ambient noises or continuous acoustic events (e.g., talking or music), filtering out these windows are the main purpose of Symptom Detector. For other non-symptom windows which contain discrete acoustic events (e.g., knocking on a desk), they will be filtered out in the next component. In what follows, we introduce the time-domain features used in Symptom Detector and illustrate how these features are used to filter out non-symptom windows when the phone works in various contexts.

Time-domain Features

Although time-domain features are not sufficient to identify each kind of respiratory symptom, they can be used to filter out many of the windows without respiratory symptoms and their calculation only consumes a little CPU and power. Three time-domain features (one frame-level feature and two window-level features) are used in Symptom Detector.

Root Mean Square (RMS): Let f denote a frame consisting of n samples and let s_i denote the normalized amplitude value (i.e., s_i is scaled from its original 16-bit binary value recorded by microphone to $[-1,1]$) of the i-th sample in f, then frame f's RMS is:

$$rms(f) = \sqrt{\frac{\sum_{i=1}^{n} {s_i}^2}{n}}$$

RMS [10] measures the energy contained in an acoustic frame and the following two window-level features are calculated based on it.

Above α-Mean Ratio (AMR): Let f_i denote the i-th frame in a window w consisting of m frames and given parameter α, w's AMR is calculated as:

$$amr(\alpha, w) = \frac{\sum_{i=1}^{m} ind[rms(f_i) > \alpha \cdot \overline{rms}(w)]}{m}$$

where $\overline{rms}(w)$ is the mean RMS of window w and $ind()$ is the indicator function which returns 1 if its argument is true and 0 otherwise.

AMR measures the ratio of the high-energy frames in a window and parameter α is used together with the window's mean RMS to set a threshold for distinguishing high-energy frames from low-energy frames. Since in indoor environment (e.g., office or home), acoustic event frames usually contain much more energy than ambient noise frames, in a window with discrete acoustic event, when α is set to close to 1, AMR approximately reflects the proportion of the event frames in the window. Given an appropriate α, windows containing discrete acoustic events, continuous acoustic events and ambient noises will return different AMR values and thus this feature can be used to sift out windows with discrete acoustic events. In SymDetector, the default value of α is set to be 0.5.

Average of Top k RMSs (ATR): Let f_i denote the frame with the i-th largest RMS in window w. Considering the top k RMSs, ATR is calculated as:

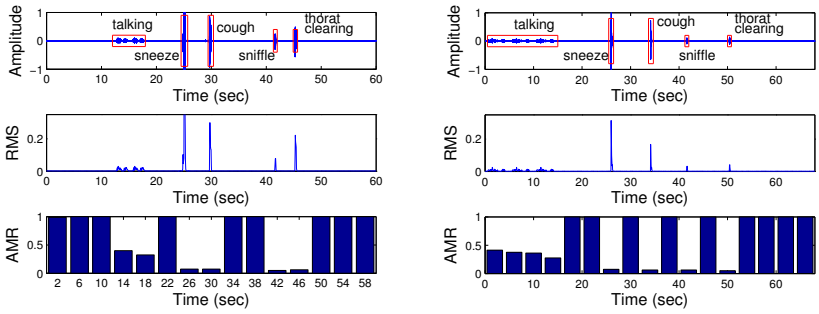
$$atr(k, w) = \frac{\sum_{i=1}^{k} rms(f_i)}{k}$$

ATR measures the average RMS of the first k frames with the most energy. It is used to discern windows containing high-energy events from windows containing low-energy events. As shown in Figure 2, since more than 95% of the respiratory symptoms last longer than 0.1 seconds (i.e., 2 frames), k is set to 2 in SymDetector.

Adaptive Symptom Detection

Based on the extracted features, two steps are used to filter out as many non-symptom windows as possible.

First, AMR is used to capture the windows with discrete acoustic events. As shown in Figure 3, for an ambient noise window, each frame has similar energy and thus when α is



(a) On the desk

(b) In the pocket

Figure 3: The RMS and AMR values of talking and flu symptoms when a phone is put on the desk and in the pocket respectively.

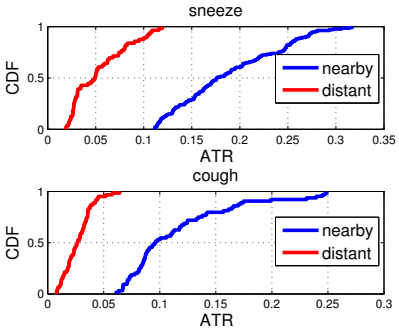


Figure 4: CDF of sneeze window ATR and cough window ATR when a phone is put close to the user and far away from the user respectively.

set to 0.5, its AMR is close to 1. For a window containing discrete acoustic event (e.g., any respiratory symptom, shutting the door, knocking on the desk), its AMR is relatively small since there are only a few number of acoustic event frames in the window, whose RMS values are much larger than those of the ambient noise frames. Similarly, the AMR of a window containing continuous acoustic event (e.g., talking or music) is smaller than 1 but larger than that of the discrete acoustic event window. Specifically, a talking window has an AMR of 0.3 to 0.5 (shown in Figure 3) since the voiced frames occupy 30% to 50% in a fluent speech [20]. Comparing Figure 3b with Figure 3a, although the energy contained in the sampled event frames decreases when a smartphone is put in the pants pocket, for the windows in the same category (i.e., ambient noise window, discrete acoustic event window or continuous acoustic event window), their AMR values do not change very much. Therefore, AMR is a robust window-level feature to classify windows into different categories regardless of the contexts of the phone.

Second, since a user may spend much of his/her time using the smartphone in a public area (e.g., offices, classes), acoustic events made by other people around him/her may also be captured by the phone. In order to filter out windows with these events, ATR is used after the first step. According to the laws of acoustic wave propagation, an acoustic wave will lose more energy when it propagates further. Therefore, in the smartphone, the recorded acoustic events made by the user who is much closer to the phone will have more energy than those made by the people around him/her. ATR reflects the energy contained in an event, thus it can be used by a phone to discern nearby events from distant events. As shown in Figure 4, the ATR of a distant event window is much smaller than that of a nearby event window, and thus the distant event window can be filtered out by an ATR threshold γ.

However, γ should not be set to a fixed value since the energy contained in an acoustic event captured by a phone will be different under various contexts. For example, the ATR of a window containing a user's sneeze captured when the phone is put in his/her pocket may not surpass the threshold γ which is designed based on acoustic samples collected when the phone is put on the desk, but this window should not be discarded as a window containing a sneeze made by others. Thus γ should be set adaptively to cope with different contexts. Since a talking window can be detected by using AMR (shown in Figure 3), we use it to determine γ. When a window w is classified as a talking window, its mean RMS $\overline{rms}(w)$ is used to update the mean RMS of all the talking windows $\overline{rms}$ as follows:

$$\overline{rms} = \overline{rms} + \beta \cdot [\overline{rms}(w) - \overline{rms}] \quad (1)$$

And then the ATR threshold γ is calculated as:

$$\gamma = \eta \cdot \overline{rms} \quad (2)$$

In SymDetector, β and η are set to 0.5 and 1.2 respectively based on our preliminary experimental results. According to Equation 1 and 2, γ will be adaptively changed based on the variation of the contexts, so Symptom Detector can sift out windows containing a user's respiratory symptoms robustly. Figure 5 shows the ATR of the windows containing respiratory symptoms made by a user (denoted as *user*) and people around him (denoted as *others*) in three weeks in different phone contexts. The phone was put on the desk for the first week, in the user's pants pocket for the second week and in his backpack for the third week. As can be seen, although the ATR of the window containing a certain type of respiratory symptom varies in different phone contexts, the ATR threshold also changes adaptively, which makes Symptom Detector be able to discern respiratory symptoms made by the *user* from *others*. As shown in Figure 5, when the phone is put on the desk, all the respiratory symptoms made by the *user* can be sifted out. For the respiratory symptoms made by *others*, except for a few sneezes, all the others can be filtered out correctly. When the phone is put in the user's pants pocket or backpack, except for a few sniffles and throat clearing symptoms, all the other symptoms made by the *user* can be dis-

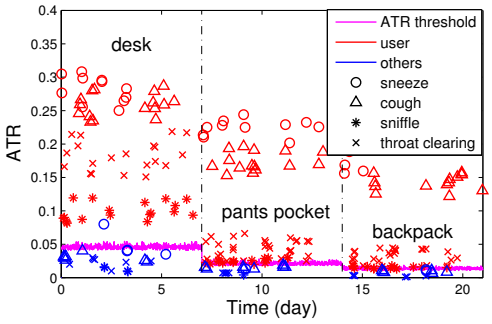


Figure 5: ATR of respiratory symptoms and the adaptive threshold in different phone contexts (The phone is put on the desk from day 1 to day 7, in the user's pants pocket from day 8 to day 14 and in the user's backpack from day 15 to day 21).

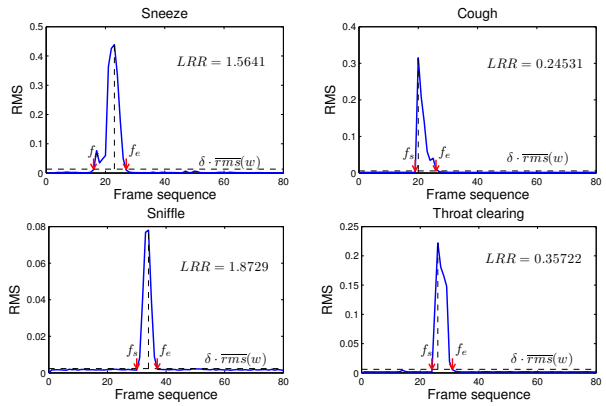


Figure 6: The calculation of SL and LRR for different types of symptoms.

cerned from those made by *others* through the ATR threshold. When a window is fed to Symptom Classifier, the threshold γ, which reflects the current signal intensity level, will also be recorded for further processing.

Symptom Classifier

In Symptom Detector, most noise windows and continuous acoustic event windows are filtered out. However, since only a few features are used there, some windows which contain other discrete acoustic events rather than respiratory symptoms (e.g., shutting the door, knocking on the desk) will also be preserved. In order to classify the preserved windows into different categories (i.e., sneeze, cough, sniffle, throat clearing and non-symptoms), in Symptom Classifier more features are extracted and multi-level classifiers are designed.

Feature Extraction

Both time-domain and frequency-domain features are extracted in Symptom Classifier.

Symptom Length (SL): As shown in Figure 6, SL measures the length of frame set F_e ($F_e = \{f_s, f_{s+1}, ..., f_e\}$), which is the largest continuous frame set covered by the accoustic event. For a discrete acoustic event in window w, to decide its F_e, the maximum RMS frame f_m is located and put into F_e initially, then the frames before and after this frame are continuously added into F_e until the frames whose RMS values are less than $\delta \cdot \overline{rms}(w)$ (δ is set to 0.5 in SymDetector) are met.

As observed in Figure 2, most of the symptoms last for 0.1 to 0.6 seconds (i.e., 2 to 12 frames), and thus SL can be used to discern the non-symptoms whose lengths are out of this range. Also, obtaining frame set F_e will save CPU and power when calculating the resource-consuming frequency-domain features since only frames in F_e, instead of frames in the entire window, need to be considered.

Left to Right Ratio (LRR): LRR measures the ratio of the area covered by the frames from f_s to f_m to the area from f_m to f_e. As shown in Figure 6, sneeze and sniffle's LRR values are larger than 1, while cough and throat clearing's LRR values are less than 1.

Relative Maximum RMS (RMR): As shown in Figure 3 and Figure 6, sneeze and cough contain much more acoustic energy than sniffle and throat clearing. RMR is used to reflect this difference and it is calculated as:

$$rmr(w) = \max_{f \in w} \frac{rms(f)}{\gamma}$$

where γ is the ATR threshold when window w is processed in Symptom Detector. Instead of absolute RMS, relative RMS is used to avoid the affect caused by different contexts.

Zero Crossing Rate (ZCR): Let $sgn()$ denote the sign function which returns 1 for a positive argument, 0 for 0 and -1 for a negative argument. Then frame f's ZCR [29] is:

$$zcr(f) = \frac{\sum_{i=2}^{n} |sgn(s_i) - sgn(s_{i-1})|}{2(n-1)}$$

ZCR is a good feature to detect percussive sounds. In Symptom Classifier, the mean and variance of ZCR in F_e are used to discern the non-symptoms whose ZCR values are out of the range of the ZCR values calculated from respiratory symptoms.

Spectral Centroid (SC): Let p_i ($i = 1, 2, ..., N$) denote the normalized magnitude of the i-th frequency bin obtained by using Fast Fourier Transform (FFT) on frame f. f's SC [16] is calculated as:

$$sc(f) = \frac{\sum_{i=1}^{N} i \cdot p_i^2}{\sum_{i=1}^{N} p_i^2}$$

SC measures the centroid of the spectral energy distribution.

Bandwidth: Following the calculation of SC, f's Bandwidth [16] bw is calculated as:

$$bw(f) = \frac{\sum_{i=1}^{N} (i - sc(f))^2 \cdot p_i^2}{\sum_{i=1}^{N} p_i^2}$$

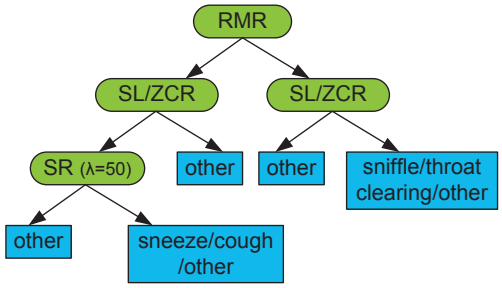


Figure 7: The coarse classifier.

Bandwidth measures the flatness of the FFT spectrum.

λth-percentile Spectral Rolloff (SR): Given parameter λ, f's SR is calculated as:

$$sr(\lambda, f) = min(h | \sum_{i=1}^{h} p_i^2 > \frac{\lambda}{100} \sum_{i=1}^{n} p_i^2)$$

SR indicates the frequency bin below which it contains λ percent of the total spectral energy. It reflects how a frame's spectral energy distributes. For example, SR of a frame whose energy mostly concentrates on low-frequency band will be small even for a large λ and SR of a high-frequency frame will be large even for a small λ. In SymDetector, λ is set to 10, 50 and 90 when this feature is extracted.

The mean and variance of these spectral features of the frames in F_e are calculated as window-level features.

Multi-level Classification

After extracting the above features from acoustic event windows, a classifier can be trained to classify respiratory symptoms. However, although most ambient noise windows and continuous acoustic event windows are filtered out in Symptom Detector, many windows which do not actually contain respiratory symptoms are still preserved (e.g., shutting the door, knocking on the desk, turning the book). Since in people's daily lives, these events occur much more than respiratory symptoms, directly using a classifier on all the windows preserved in Symptom Detector will cause the *Class Imbalance Problem* [14]. Therefore, a coarse classifier is designed in Symptom Classifier to filter out as many non-symptom windows as possible and then Support Vector Machine (SVM) is used to classify all the respiratory symptoms.

As shown in Figure 7, in the coarse classifier, RMR is used to classify the events as two categories due to the higher energy level of sneeze and cough than that of sniffle and throat clearing (shown in Figure 3, Figure 5 and Figure 6). Then SL and ZCR are used since they are time-domain features and many non-symptoms can be discerned by using them. For example, as observed in our experimental data, the sound made when one puts his coffee cup on the desk has low SL and the sound of shutting a door has low ZCR. Lastly, SR with λ = 50 is used to filter out high-frequency events in the category of sneeze and cough since both of them do not contain many high-frequency energy.

After identifying most of the non-symptom windows and classifying the remaining as two categories in coarse classifier, a finer classifier is used on both categories to classify the respiratory symptoms. In machine learning, SVM is shown to be an effective supervised learning technique and has been used in various classification problems [11, 5, 32]. Therefore, we use SVM as the second-level classification in SymDetector. However, SVM is originally designed for binary classification. In our system, after running the coarse classifier, three types of sounds need to be classified in each category (i.e., sneeze, cough and non-symptom in one category; sniffle, throat clearing and non-symptom in the other category). Thus, the basic SVM technique needs to be extended to classify multiple types of sounds. *One-against-all* [2] and *one-against-one* [9] are two well-known strategies of using binary SVM for multiclass classification. For k classes, although the number of binary SVMs constructed by one-against-one (i.e., $k(k-1)/2$) is larger than that of one-against-all (i.e., k), one-against-one yields higher classification accuracy in general [11]. Since in our classification problem, each category only has 3 types of sounds, we use one-against-one strategy and use Radial Basic Function (RBF) as the kernel function when training the binary SVMs.

There are some other classification schemes which can also be used for symptom classification, such as Gaussian Mixture Models (GMM) [8], k-nearest Neighbors (KNN) [19], Hidden Markov Models (HMM) [1] and Random Forest (RF) [15]. HMM and RF are used in [1] and [15] respectively to detect coughs, and we will compare our system with theirs in the performance evaluations.

Symptom Recorder

The classified respiratory symptoms are recorded in this component. Since overlapped windows are used in SymDetector, a respiratory symptom may be recorded twice. To remove redundancy, f_s and f_e, which are obtained when extracting SL, are recorded in terms of system time, and if two recorded respiratory symptoms are interleaved, the one with shorter SL will be discarded. The recorded respiratory symptoms can be accessed by the user locally or shared with medical researchers with the permission of the user.

IMPLEMENTATION

SymDetector is implemented on Samsung's smartphone Galaxy S3, using Android OS 4.2.2. Acoustic samples are continuously read by a sampling thread from the phone's built-in microphone at 16 KHz. After 64K samples (i.e., 4 seconds) are obtained, they are fed to a processing thread. The sampling thread will start building the next window from the 3*rd* second of the previous one so that these two windows will have 1-second overlap. The processing thread segments the window into frames of 800 samples and calculates each frame's RMS to get the window's AMR and ATR. Symptom Detector is implemented to decide if this window should be discarded or preserved. If it is detected as a talking window, ATR threshold γ will be updated based on Equation 1 and 2 before the window is discarded. If the window is preserved, Symptom Classifier is implemented to classify it into one of the categories (i.e., sneeze, cough, sniffle, throat clearing or

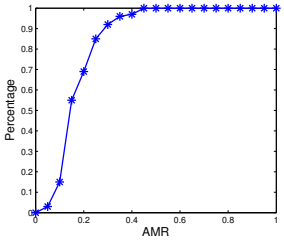


Figure 8: The percentage of respiratory symptoms that are recorded when differen AMR thresholds are used based on the preliminary dataset.

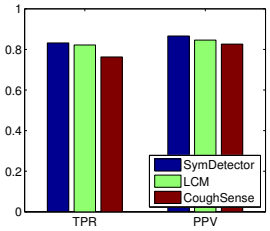


Figure 9: Cough detection results of *SymDetector*, *LCM* and *CoughSense*.

User	Days	Working conditions	Sneeze	Cough	Sniffle	Throat clearing	Non-symptoms
1	28	office	41	44	61	91	814
2	24	office	55	64	15	42	726
3	17	office/home	36	27	14	4	660
4	16	office/home	40	23	12	3	562
5	15	office	26	7	5	16	413
6	14	office/home	10	21	48	65	632
7	14	office/home	14	3	12	4	465
8	14	office	13	2	6	3	357
9	13	office	8	36	10	13	437
10	7	office/home	14	78	1	28	272
11	7	office	10	7	1	3	263
12	7	office	12	3	0	7	176
13	7	office	8	2	0	0	230
14	7	office	18	0	2	5	216
15	7	office/home	7	1	0	2	325
16	7	office	17	2	0	0	169
total	204	-	329	320	187	286	6717

Table 1: Overview of the experiment and collected data.

non-symptoms). The occurrence time and features of the classified event are recorded by a recording thread. Then the processed window is discarded and the processing thread ends. SymDetector does not keep any raw acoustic data and it only buffers 4-second data during the process. The system time when a window is fetched is recorded to infer the starting and ending time of the respiratory symptom if it exists in the window.

In the processing thread, if a window is preserved after Symptom Detector, all its time-domain and frequency-domain features mentioned before will be extracted. Then, a coarse classifier is used to classify the window into one of the two categories as shown in Figure 7, and 6 binary SVMs are implemented (3 for each category) to further classify the window as one of the symptoms or non-symptom. In order to save the smartphone's CPU and power, we use Libsvm [4] to train the 6 SVMs offline, and then the support vectors and the coefficients are provided to the processing thread for classification.

SymDetector is easy to use since it is designed to work in an unobtrusive way. After the program is started, the user can use the phone just in his/her normal pattern and no special instructions need to be followed. SymDetector will stop when the user is answering or making a phone call and resume after that. We provide the user an access to his/her recorded data, so the user can either track his/her respiratory symptoms during a certain period locally or upload his/her respiratory symptom data to a trustworthy server.

PERFORMANCE EVALUATIONS

In this section, we evaluate the performance of SymDetector based on the data collected in real experiments.

Experimental Setup

We have two datasets: a preliminary dataset collected from 5 users which is used to study the features extracted from respiratory symptoms and determine the parameters used in our system, and a dataset collected from 16 users to evaluate the performance of our system.

The preliminary acoustic dataset is collected from 5 users. Each user was given a Galaxy S3 phone, which was carried by them and recorded all the acoustic data around them from 9am to 12pm every experimental day as *wav* files. The experiment lasted 7 days and after the experiment, a total length of 105-hour audio clips were collected. We asked the users to listen to the audio files to label all the sound events (e.g., conversations, respiratory symptoms made by themselves and others). The labeled data is then used to study the features extracted from respiratory symptoms (e.g., the distribution of symptom length in Figure 2) and further to determine the parameters used in SymDetector (e.g., α, β and η used in Symptom Detector).

This preliminary dataset is relatively small. This is because increasing the number of users and recording their acoustic data and then listening to them to label the ground-truth symptoms is very difficult and impractical. First, it is hard to recruit users to participate in the experiment because they may feel uncomfortable when knowing that all their acoustic data (including their daily conversation) will be recorded. Second, it will cost much time and labor to get the ground truth because all the audio files must be played and listened in order to label respiratory symptoms and other acoustic events from the collected audio data. Third, recording all the acoustic data as *wav* or other playable audio files will consume much stor-

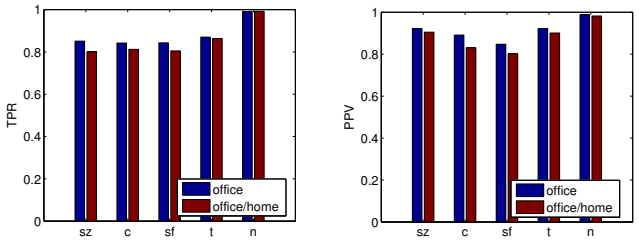
age and this will make it difficult to conduct the experiment continuously for a long time because of the limited storage in smartphones.

To overcome these problems, we design a data collection scheme to reduce the audio data to be recorded without missing respiratory symptoms. Since the discrete acoustic events (e.g., respiratory symptoms) can be distinguished from the ambient noises and continuous acoustic events (e.g., conversations) by using AMR extracted in Symptom Detector (shown in Figure 3), we can reduce the audio data by only recording the discrete acoustic events. In our experiment, SymDetector is modified not to detect respiratory symptoms directly. Instead, it records the 4-second window audio clips and the cross-validated training and testing are done offline. For each 4-second window sampled in Audio Sampler, its AMR value is calculated in real time. If the AMR is below a threshold, this window will be recorded as a *wav* file to provide ground truth; otherwise, it will be discarded. As shown in Figure 8, the experiment based on the preliminary dataset indicates that all the respiratory symptoms can be safely recorded if the AMR threshold is set appropriately (it is set to 0.5 in our data collection). In the end, we collect the data from users and listen to all these 4-second audio files to label respiratory symptoms and other acoustic events manually.

Using the above scheme, we recruited 16 users and collected our second dataset. These users were asked to use the smartphone with SymDetector running for at least 6 hours a day. The numbers of respiratory symptoms and other acoustic events (called *non-symptoms*) made by each user are shown in Table 1. Among the 16 users ranging from 18 to 30 years old (6 females and 10 males), 12 are graduate students who spend most of their daily time in their offices. During the experiment, two users (user 6 and user 10) reported that they happened to catch a cold and two users (user 2 and user 4) reported that they were troubled with pollen allergy. Since SymDetector is designed for indoor environment, in our experiment, the audio data is only collected from two common indoor scenarios (i.e., office and home). As shown in column 3 of Table 1, some users only use it in their offices during the day (*office*) and the others are asked to use SymDetector at both offices and home (*office/home*). As can been seen from Table 1, our design largely reduces the audio data recorded for ground truth. For example, even for a 28-day experiment (user 1), only 1051 audio clips (i.e., 136MB audio data) are recorded. This also reduces the time spent on labeling the ground-truth events. We only spent less than 15 hours to label all the respiratory symptoms monitored in 204 days.

After gathering data from users and labeling all the acoustic events, the classification performance of SymDetector is cross validated under different conditions. *Leave-one-user-out* strategy is used in our evaluation for cross validation across users. Each time, data collected from 15 users is used as training data to train the coarse classifier and SVM classifier, and data gathered from the remaining user is used for testing. This process is repeated so that each user's data is used exactly once as the testing data.

Overall Performance



(a) TPR (b) PPV

Figure 10: TPR and PPV of each type of acoustic event under different working conditions.

Symptom	TPR	PPV
Sneeze	0.836	0.910
Cough	0.831	0.866
Sniffle	0.824	0.827
Throat clearing	0.867	0.914
Non-symptoms	0.991	0.985

Table 2: The detection results of respiratory symptoms based on the data collected from 16 users and 204 days.

Table 2 shows the overall detection results of SymDetector in terms of *True Positive Rate* (TPR) and *Positive Predictive Value* (PPV). For a certain type of event, its TPR is defined as the ratio of the number of true positives (i.e., the events which are correctly identified as such type) to the number of actual positives (i.e., such type of events which are actually in the test set) and its PPV is defined as the ratio of the number of true positives to the sum of the number of true positives and false positives (i.e., the events which are identified as such type but actually not). As can be seen, more than 82% of respiratory symptoms are correctly classified and 99.1% of non-symptoms are detected. The PPV value of respiratory symptoms are larger than 82%, indicating that only a few of acoustic events are misclassified as a certain type of respiratory symptom. Comparing with the other types of respiratory symptoms, throat clearing is detected more accurately and has fewer false positives due to its low-frequency characteristic. The TPR and PPV of sniffle are not as large as those of the other respiratory symptoms. This is because the duration of sniffle is very short and its frames contain little energy, which makes it difficult to be distinguished from some non-symptoms. Sneeze contains much more energy than many other acoustic events occurred in a user's daily life. Therefore, only a small number of acoustic events are falsely classified as sneeze and it has a large PPV value (91.0%).

Since there is no similar work on detection of all these four types of respiratory symptoms, we only consider cough and compare SymDetector with two cough detection systems (denoted as *LCM* [1] and *CoughSense* [15] respectively) by using the dataset collected from 16 users. *LCM* uses Hidden Markov Models to detect coughs based on the audio data collected from a microphone worn around a user's neck. *Cough-*

Component	CPU usage	Power consumption
Audio Sampler	1.67%	621.72mJ
Symptom Detector	1.13%	130.28mJ
Symptom Classifier	2.73%	326.35mJ
Symptom Recorder	0.45%	105.44mJ

Table 3: CPU usage and power consumption of each component in SymDetector when a respiratory symptom window is processed.

System	CPU usage		Power consumption	
	Ambient noise	Cough	Ambient noise	Cough
SymDetector	2.91%	5.72%	0.73J	1.25J
CoughSense	10.36%	13.28%	1.57J	1.93J

Table 4: CPU usage and power consumption of *SymDetector* and *CoughSense* when processing an ambient noise window and a cough window.

Sense uses a smartphone to record the audio data and uses Random Forest to detect coughs. Both these schemes are shown to be effective in detecting coughs. However, in their experiments, the audio data is collected without considering different phone contexts. Therefore, we implemented these two schemes and compared them with SymDetector. As shown in Figure 9, *SymDetector* has larger TPR and PPV than *LCM* and *CoughSense*. This is because both time-domain and frequency-domain features are used in *SymDetector* and these features are effective in cough detection, while in *LCM* and *CoughSense* only frequency-domain features are used. Also, *SymDetector* is designed to work in different contexts, while *LCM* and *CoughSense* can only work when the microphone is put in a specific position. Our experimental data is collected in various contexts, and thus *SymDetector* performs better than the other two systems.

Working Conditions

As illustrated before, two groups of users are asked to test SymDetector under different working conditions. Comparing with *office*, in *office/home*, the phone's working condition is more complicated, and thus more non-symptoms are recorded when a phone works in *office/home*. As observed in Table 1, for a 7-day experiment, user 10 and 15 (*office/home*) collected more non-symptoms than the others (*office*). To compare SymDetector's performances under these two working conditions, data collected in *office* and *office/home* is analyzed respectively and Figure 10 shows their detection results on sneeze, cough, sniffle, throat clearing and non-symptoms (denoted as *sz*, *c*, *sf*, *t* and *n* respectively). As observed in Figure 10a, more than 80% of the respiratory symptoms in each type are correctly detected under both working conditions. Comparing Figure 10a with Figure 10b, for each type of respiratory symptom, its TPR almost remains the same under different working conditions, but its PPV is higher in *office* than in *office/home*. This is because more different types of non-symptoms occur in *office/home* than in *office*, which results in more non-symptoms being misclassified as symptoms.

CPU Usage and Power Consumption

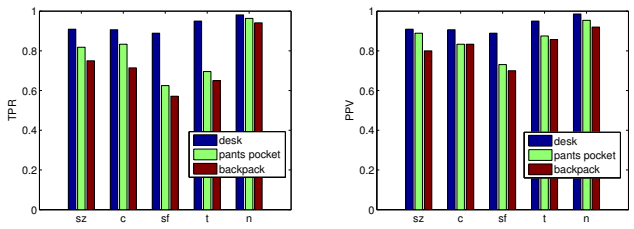
Table 3 shows the average CPU usage and power consumption of each component in SymDetector when a respiratory symptom window is processed. As can be seen, Symptom Classifier consumes more CPU (2.73%) than any other component. This is because it extracts many time-domain and frequency-domain features and uses multi-level classifiers to detect respiratory symptoms. Comparatively, Symptom Detector only extracts three time-domain features and thus it consumes little CPU (1.13%) and power (130.28 mJ). In SymDetector, low-power Symptom Detector is used to process all the windows and only a few number of windows which contain discrete acoustic events are processed by the resource-consuming Symptom Classifier.

Since *CoughSense* is also designed on smartphone, we compare the average CPU usage and the power consumption of *SymDetector* and *CoughSense* when different windows are processed, and the results are shown in Table 4. As can be seen, since *CoughSense* needs to extract frequency-domain features for all the frames in a window, it consumes more CPU and power than *SymDetector* in processing either an ambient noise window or a cough window. Our experimental result shows that on average, SymDetecor consumes 803.53 mJ for processing a 4-second window and it can work for more than 20 hours on a fully charged phone.

DISCUSSIONS

In the design of SymDetector, we consider users' privacy, power consumption and phone context to make it practical to detect and record users' respiratory symptoms for a long time. Although users with respiratory diseases may have many symptoms and the symptoms detected in our current work may not be sufficient to exactly infer whether a user has got certain disease or not, detecting these 4 types of sound-related symptoms, which are commonly observed in many respiratory diseases, will provide useful information for the medical researchers. For example, these symptoms can be collected in some surveillance systems to monitor the spread of infectious diseases like flu [6] and may potentially help with infectious disease containment [31]. Also, continuously monitoring a specific user's symptoms and comparing them with his/her historical information will help to detect some diseases in an early stage.

Although SymDetector is designed to work in different phone contexts, its detection result may vary as the context changes. In order to evaluate SymDetector's performance in various phone contexts, we collected three more weeks' data from user 1. For each week, we asked user 1 to put the phone on the desk, in his pants pocket and in his backpack (the backpack was left close to the user during his working) respectively (denoted as *desk*, *pants pocket* and *backpack* respectively). Due to the small scale of our dataset, we train the classifiers based on user 1's acoustic data collected in Table 1 and show the detection results in Figure 11. As observed in Figure 11a, although the TPR values of all four types of respiratory symptoms in *pants pocket* and *backpack* are smaller than those of *desk*, more than 75% of sneezes and coughs and more than



(a) TPR (b) PPV

Figure 11: TPR and PPV of each type of acoustic event in different phone contexts.

55 % of sniffles and throat clearing symptoms in *pants pocket* and *backpack* are correctly detected. In *pants pocket* and *backpack*, comparing with sneezes and coughs, less sniffles and throat clearing symptoms are correctly detected because these two types of symptoms contain less acoustic energy, which makes them more likely to be filtered out in Symptom Detector. As shown in Figure 11b, similar to TPR, for each type of respiratory symptom, its PPV in *pants pocket* and *backpack* is smaller than that of *desk*. However, comparing Figure 11b with Figure 11a, the PPV value difference between *pants pocket* (or *backpack*) and *desk* for a certain type of respiratory symptom is smaller than the corresponding TPR value difference. This is because in the extracted features, except for RMR, all the other features are not related to the amount of energy contained in the acoustic events. Although the energy contained in each respiratory symptom in *pants pocket* and *backpack* is less than that of *desk*, it is still hard for a non-symptom to be misclassified as a respiratory symptom.

The current SymDetector is designed to work in indoor environment, where the ambient noise is relatively simple and invariant. However, in outdoor environment, the ambient noise is much more complicated than that indoor and noises like the sounds of birds, winds or vehicles may be collected in the acoustic data as well. These noises are hard to be filtered out in our current system, which makes the symptom detection even harder since the symptoms may be overlapped with these unpredictable ambient noises. While users will spend much of their time indoor, they may also go outdoor sometimes for walk, game or meal. As our future work, we will study the ambient noise in outdoor environment and make SymDetector work both indoor and outdoor.

RELATED WORK

By leveraging smartphones, many works have been proposed to provide users health related applications. Shahriar *et al.* [26] equipped a pair of sensors in the smartphone's earphone to monitor the user's heart rate and suggest music for the user to maintain the heart rate. Keally *et al.* [12] combined TinyOS motes and Android smartphones together to build *Practical Body Networking* (PBN) for recognizing people's daily activities. By processing the sensor data continuously read from a smartphone's accelerometer, Agata and Robert [3] designed an algorithm for walk detection and step counting. Having the similar goal with these research, we also exploit the ubiquity of smartphones to obtain users' health information.

As a low-cost and common sensor on all kinds of mobile phones, the microphone has been exploited in many applications. In [21], Hong *et al.* designed *StressSense* to evaluate a user's stress level by analyzing his/her speech. However, to collect the user's speech, the phone has to be attached to the specific part of the user's body. *SoundSense* [22], which uses both supervised and unsupervised learning techniques to detect and classify accoustic events occurred in one's daily life, considers various contexts of the phone, but it consumes lots of power since it needs to extract many frequency-domain features for each frame. By analyzing the acoustic data sensed during a user's sleep, Tian *et al.* [10] designed *iSleep* to evaluate the user's sleeping quality in terms of the number of his/her body movement, cough and snore at night. However, their techniques cannot be directly applied in SymDetector since they assume that there are no other acoustic events during one's sleep except body movement, cough and snore, which is quite different from the environment where SymDetector works.

Audio based systems have been designed to collect people's health information in many previous works. In [23] and [24], HMM based schemes are proposed to detect cough from continuous audio record. In [28] and [34], audio data is collected to analyze the lung sounds. However, these systems cannot work in real time. In [27], a mobile sensing system is proposed to detect various non-speech body sounds, but the users have to wear specific sensors. *Random Forest* (RF) classifier is implemented in [15] to detect a user's cough. In [35], *BodyScope* is designed to detect 12 non-verbal sounds including cough. However, in these systems, the user has to wear the phone (or microphone) around his/her neck. Also, the system in [15] consumes much CPU and power since it needs to calculate audio spectrogram for the entire audio sequence.

CONCLUSION

In this paper, we designed SymDetector, a smartphone based application which can unobtrusively detect a user's acoustic events related to respiratory symptoms, including sneeze, cough, sniffle and throat clearing. Several practical issues, such as users' privacy concerns about their acoustic data, resource constraints of the smartphone and different contexts of the smartphone, are considered in developing SymDetector. We have implemented SymDetector on Galaxy S3 and evaluated its performance in real experiments involving 16 users and 204 days. The experimental results show that SymDetector can detect the respiratory symptoms with high accuracy under various conditions.

ACKNOWLEDGMENTS

We would like to thank all the anonymous reviewers for their insightful comments and helpful suggestions. This work was supported in part by the National Science Foundation (NSF) under grant CNS-1218597, CNS-1320278, and CNS-1421578.

REFERENCES

1. Birring, S., Fleming, T., Matos, S., Raj, A., Evans, D., and Pavord, I. The leicester cough monitor: preliminary validation of an automated cough detection system in chronic cough. *European Respiratory Journal 31*, 5 (2008), 1013–1018.
2. Bottou, L., Cortes, C., Denker, J. S., Drucker, H., Guyon, I., Jackel, L. D., LeCun, Y., Muller, U. A., Sackinger, E., Simard, P., et al. Comparison of classifier methods: a case study in handwritten digit recognition. In *International Conference on Pattern Recognition*, IEEE Computer Society Press (1994), 77–77.
3. Brajdic, A., and Harle, R. Walk detection and step counting on unconstrained smartphones. In *ACM UbiComp* (2013).
4. Chang, C.-C., and Lin, C.-J. Libsvm: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST) 2*, 3 (2011), 27.
5. Chechik, G., Ie, E., Rehn, M., Bengio, S., and Lyon, D. Large-scale content-based audio retrieval from text queries. In *ACM Int'l Conf. on Multimedia information retrieval* (2008).
6. Chunara, R., Aman, S., Smolinski, M., and Brownstein, J. S. Flu near you: an online self-reported influenza surveillance system in the USA. *Online Journal of Public Health Informatics 5*, 1 (2013).
7. Daisey, J. M., Angell, W. J., and Apte, M. G. Indoor air quality, ventilation and health symptoms in schools: an analysis of existing information. *Indoor air 13*, 1 (2003), 53–64.
8. Dhanalakshmi, P., Palanivel, S., and Ramalingam, V. Classification of audio signals using aann and gmm. *Applied Soft Computing 11*, 1 (2011), 716–723.
9. Friedman, J. Another approach to polychotomous classifcation. *Dept. Statist., Stanford Univ., Stanford, CA, USA, Tech. Rep* (1996).
10. Hao, T., Xing, G., and Zhou, G. isleep: unobtrusive sleep quality monitoring using smartphones. In *ACM SenSys* (2013).
11. Hsu, C.-W., and Lin, C.-J. A comparison of methods for multiclass support vector machines. *Neural Networks, IEEE Transactions on 13*, 2 (2002), 415–425.
12. Keally, M., Zhou, G., Xing, G., Wu, J., and Pyles, A. Pbn: towards practical activity recognition using smartphone-based body sensor networks. In *ACM SenSys* (2011).
13. Kraman, S. S., Wodicka, G. R., Pressler, G. A., and Pasterkamp, H. Comparison of lung sound transducers using a bioacoustic transducer testing system. *Journal of Applied Physiology 101*, 2 (2006), 469–476.
14. Kubat, M., Matwin, S., et al. Addressing the curse of imbalanced training sets: one-sided selection. In *ICML*, vol. 97, Nashville, USA (1997), 179–186.
15. Larson, E. C., Lee, T., Liu, S., Rosenfeld, M., and Patel, S. N. Accurate and privacy preserving cough sensing using a low-cost microphone. In *ACM UbiComp* (2011).
16. Li, D., Sethi, I. K., Dimitrova, N., and McGee, T. Classification of general audio data for content-based retrieval. *Pattern recognition letters 22*, 5 (2001), 533–544.
17. Li, Q., and Cao, G. Privacy-preserving participatory sensing. *IEEE Communication Magazine*, to appear.
18. Li, Q., and Cao, G. Providing privacy-aware incentives for mobile sensing. In *IEEE Percom* (2013).
19. Liao, W.-H., Wen, J.-Y., and Kuo, J.-H. Streaming audio classification in smart home environments. In *Pattern Recognition (ACPR), 2011 First Asian Conference on*, IEEE (2011), 593–597.
20. Liu, Z., Wang, Y., and Chen, T. Audio feature extraction and analysis for scene segmentation and classification. *Journal of VLSI 20*, 1-2 (1998), 61–79.
21. Lu, H., Frauendorfer, D., Rabbi, M., Mast, M. S., Chittaranjan, G. T., Campbell, A. T., Gatica-Perez, D., and Choudhury, T. Stresssense: Detecting stress in unconstrained acoustic environments using smartphones. In *ACM UbiComp* (2012).
22. Lu, H., Pan, W., Lane, N. D., Choudhury, T., and Campbell, A. T. Soundsense: scalable sound sensing for people-centric applications on mobile phones. In *ACM MobiSys* (2009).
23. Matos, S., Birring, S. S., Pavord, I. D., and Evans, D. H. Detection of cough signals in continuous audio recordings using hidden markov models. *Biomedical Engineering, IEEE Transactions on 53*, 6 (2006), 1078–1083.
24. Matos, S., Birring, S. S., Pavord, I. D., and Evans, D. H. An automated system for 24-h monitoring of cough frequency: the leicester cough monitor. *Biomedical Engineering, IEEE Transactions on 54*, 8 (2007), 1472–1479.
25. McGuinness, K., Kelsall, A., Lowe, J., Woodcock, A., and Smith, J. Automated cough detection: a novel approach. *Am J Respir Crit Care Med 175* (2007), A381.
26. Nirjon, S., Dickerson, R. F., Li, Q., Asare, P., Stankovic, J. A., Hong, D., Zhang, B., Jiang, X., Shen, G., and Zhao, F. Musicalheart: A hearty way of listening to music. In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*, ACM (2012), 43–56.
27. Rahman, T., Adams, A. T., Zhang, M., Cherry, E., Zhou, B., Peng, H., and Choudhury, T. Bodybeat: A mobile system for sensing non-speech body sounds. In *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*, ACM (2014), 2–13.

28. Reichert, S., Gass, R., Brandt, C., and Andrès, E. Analysis of respiratory sounds: state of the art. *Clinical medicine. Circulatory, respiratory and pulmonary medicine 2* (2008), 45.

29. Saunders, J. Real-time discrimination of broadcast speech/music. In *IEEE Int'l Conf. on Acoustics, Speech and Signal Processing (ICASSP)* (1996).

30. Smith, J., and Woodcock, A. New developments in the objective assessment of cough. *Lung 186*, 1 (2008), 48–54.

31. Sun, X., Lu, Z., Zhang, X., Salathé, M., and Cao, G. Targeted vaccination based on a wireless sensor system. In *IEEE Percom* (2015).

32. Temko, A., Macho, D., and Nadeu, C. Fuzzy integral based information fusion for classification of highly confusable non-speech sounds. *Pattern Recognition 41*, 5 (2008), 1814–1823.

33. Temko, A., and Nadeu, C. Classification of acoustic events using svm-based clustering schemes. *Pattern Recognition 39*, 4 (2006), 682–694.

34. Whittaker, A., Lucas, M., Carter, R., and Anderson, K. Limitations in the use of median frequency for lung sound analysis. *Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine 214*, 3 (2000), 265–275.

35. Yatani, K., and Truong, K. N. Bodyscope: a wearable acoustic sensor for activity recognition. In *ACM Ubicomp* (2012).