Darwin Phones: the Evolution of Sensing and Inference on Mobile Phones

Emiliano Miluzzo[†], Cory T. Cornelius[†], Ashwin Ramaswamy[†], Tanzeem Choudhury[†], Zhigang Liu[§], Andrew T. Campbell[†]

[†]Computer Science, Dartmouth College, Hanover, NH, USA

[§]Nokia Research Center, 955 Page Mill Road, Palo Alto, CA, USA

ABSTRACT

We present Darwin, an enabling technology for mobile phone sensing that combines collaborative sensing and classification techniques to reason about human behavior and context on mobile phones. Darwin advances mobile phone sensing through the deployment of efficient but sophisticated machine learning techniques specifically designed to run directly on sensor-enabled mobile phones (i.e., smartphones). Darwin tackles three key sensing and inference challenges that are barriers to mass-scale adoption of mobile phone sensing applications: (i) the human-burden of training classifiers, (ii) the ability to perform reliably in different environments (e.g., indoor, outdoor) and (iii) the ability to scale to a large number of phones without jeopardizing the "phone experience" (e.g., usability and battery lifetime). Darwin is a collaborative reasoning framework built on three concepts: classifier/model evolution, model pooling, and collaborative inference. To the best of our knowledge Darwin is the first system that applies distributed machine learning techniques and collaborative inference concepts to mobile phones. We implement the Darwin system on the Nokia N97 and Apple iPhone. While Darwin represents a general framework applicable to a wide variety of emerging mobile sensing applications, we implement a speaker recognition application and an augmented reality application to evaluate the benefits of Darwin. We show experimental results from eight individuals carrying Nokia N97s and demonstrate that Darwin improves the reliability and scalability of the proof-of-concept speaker recognition application without additional burden to users.

Categories and Subject Descriptors: C.3 [Special-Purpose and Application-Based Systems] Real-time and embedded systems

General Terms: Algorithms, Design, Experimentation, Human Factors, Measurement, Performance

Keywords: Mobile Sensing Systems, Distributed Machine Learning, Collaborative Inference, Mobile Phones

MobiSys'10, June 15–18, 2010, San Francisco, California, USA.

1. INTRODUCTION

The continuing need to communicate has always pushed people to invent better and more efficient ways to convey messages, propagate ideas, and share personal information with friends and family. Social-networking, for example, is the fastest growing phenomenon of the Internet era where people communicate and share content with friends, family, and acquaintances. Recently, researchers started investigating new ways to augment existing channels of communication and improve information exchange between individuals using the computational and sensing resources offered by sensor-enabled mobile phones (aka smartphones). These phones already utilize sensor data to filter relevant information (e.g., location-based services) or provide better user experiences (e.g., using accelerometer data to drive mobile phone sensing applications). However, information about user's behavior (e.g., having a conversation) and personal context (e.g., hanging out with friends) is often provided manually by the user. This naturally leads to the following thoughts: what if the available sensors are further exploited to automatically infer various aspects of a person's life in ways that have not been done before? What if the characterization of the person's *microcosmos* could be seen as a new form of communication? We believe that as sensor-enabled mobile phones become commonplace, they can be used at a personal-scale to enrich and support communication and collaboration, to measure and improve task performance, and to aide in the assessment of health and wellness.

There is a growing research effort in using mobile phones to infer information about people's behavior and their environment [46, 21, 20, 30, 29, 10, 27]. These systems typically rely on pre-trained models or classifiers, where the training data from events of interest are acquired in advance. It is often exceedingly hard to obtain a representative training set from which to build reliable classifiers (e.g., samples of an individual's voice in all possible environments). As a result classifiers tend to perform poorly. Furthermore, current approaches do not take advantage of increased sensing density offered, for example, by the cloud of mobile phones around us. This cloud represents an ensemble of in situ resources that can cooperate to boost sensing performance, make sensing more robust, and more effectively and scalably achieve a common sensing task.

With the rising interest in mobile phone sensing applications we believe there is a need to provide mechanisms that maximize the robustness, accuracy, and scalability of these applications. In this paper, we present Darwin, a novel col-

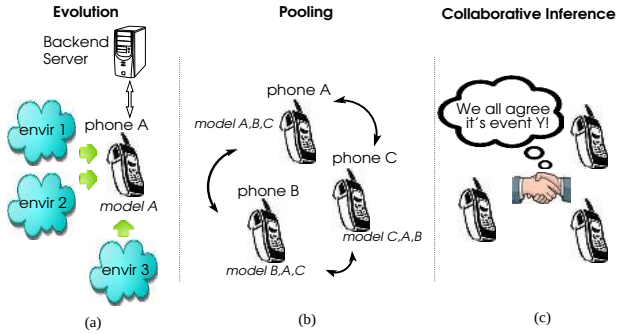


Figure 1: Darwin steps: (a) evolution, (b) pooling and (c) collaborative inference. They represent Darwin's novel evolve-pool-collaborate model implemented on mobile phones.

laborative reasoning system that is self-extending and utilizes co-located mobile devices to achieve better accuracy and scalability, at lower cost to the user. As shown in Figure 1, Darwin combines three different computational steps to achieve its goal:

Classifier evolution is an automated approach to updating models over time such that the classifiers are robust to the variability in sensing conditions common to mobile phones (e.g., phone in the pocket, in pocket bag, out of the pocket), and settings (e.g., noisy and loud environments). A fully supervised learning method, where labeled examples from different context are provided to the system, would not be practical in this case since the phone owner would continually have to provide labels any time an event is detected that is determined to be in a different setting and context. This simply does not scale and would be unacceptable to users. While self-evolving classification models techniques have been investigated in the past [54], we show the actual deployment of such techniques on a real phone based system.

Model pooling is a novel technique which is designed to answer the following question: can we reuse models that have already been built and possibly evolved on other phones? This would increase the system scalability because there would be no need to retrain classifiers for events which already have classifiers trained to recognize them. With pooling, mobile phones exchange classification models whenever the model is available from another phone, thus, allowing mobile phones to quickly expand their classification capabilities; that is, if a given mobile phone does not have a model for a person or an event, there is no need for it to create a new classifier as it can readily obtain and use another phone's model. Note, that models can be exchanged in-situ between co-located phones or from servers over the network. In either case the basic pooling process remains the same.

Collaborative inference combines the classification results from multiple phones to achieve better, more robust inference with higher confidence in the sensing result. After pooling, co-located mobile phones all have the same classifiers. At this point they can run the same inference algorithms in parallel and a final inference result can be obtained by combining the output from the different phones. This allows the system to be more robust to degradation in sensing quality experienced by some of the phones (e.g., a person carrying the phone in the pocket) and take advantage of improved sensing quality offered by others (e.g., the phone is out of the pocket near the event to be sensed).

We show the performance of Darwin by exploiting the audio modality of mobile phones, in particular, we show the benefit of applying Darwin to a speaker recognition application using audio sampled by the onboard microphone. We show the performance of the speaker recognition algorithm on the Nokia N97 [6] and Apple iPhone [1] in different settings and context when Darwin and the speaker recognition application are used by eight people.

The reason we select speaker recognition is not because we intend to design a new speaker recognition algorithm (there is a considerable amount of literature on this topic [44, 43, 26, 18, 51]), but to show how Darwin improves a mobile sensing application inference quality.

Darwin is founded on an opportunistic sensing paradigm [12], where the user is not an active participant in the sensing activity (i.e., actively taking a sensor reading). In this case, sensing happens automatically and continuously when the system determines that the sensing context is right for sensing. Darwin can be thought of as a sensing system running in the "background mode" of the mobile phone without any user intervention in actual sensing. The key contribution of our work is to show how Darwin can boost the inference accuracy of mobile sensing systems by applying *distributed computing and collaborative inference concepts* to these systems when devices come together opportunistically. We conjecture that Darwin applied to other mobile phone sensing applications and systems that use the microphone as an audio sensor [27, 10] would also see similar performance improvements because audio sensed data is sensitive to the characteristics of the environment (e.g., noise, other people speaking, etc.) and sensor context of the phone (e.g., in or out of pocket for example). In the paper, we also show how Darwin could be integrated in a mobile social networking application, a place discovery application, and a friend tagging application.

Today, mobile phone sensing applications mostly exploit the GPS, accelerometer, digital compass, and microphone sensors for personal sensing. In the future, mobile phone sensing will be societal-scale supporting a broad set of social, health and environmental applications, such as, tracking pollution, population well-being, or the spread of disease. It is also likely that more sophisticated sensors will be embedded in phones, such as, pollution and air quality sensors [20] and galvanic skin response (GSR) sensors. The sensing and inference quality of these applications is affected by many factors. Importantly, *phone sensing context*, i.e., the position of the phone on a person's body in relation to the sensed event, is challenging for these emerging applications. A phone in the pocket or bag might perform poorly when sensing air quality or audio events. Classification models are also limited by the quality of the trained data and their inability to capture different characteristics from the data in different environments. Darwin's novel **evolve-pool-collaborate model** is designed to provide a foundation for a broad family of existing and emerging sensors and applications, as shown in Figure 2. To the best of our knowledge, Darwin represents the first system implemented on a number of mobile phones that can evolve, pool, and enable cooperation providing robust, efficient, and scalable sensing.

The structure of the paper is as follows. Section 2 presents the detailed design of the Darwin system, followed in Sec-

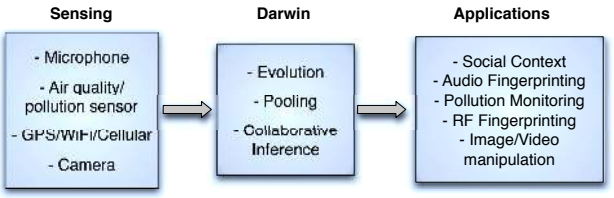


Figure 2: Examples of application domains Darwin can be applied to: social context (e.g., in conversation, in a meeting) and ambient audio fingerprinting using the microphone; pollution monitoring leveraging the phone's pollution sensor; radio fingerprinting for localization with GPS, WiFi and cellular triangulation; and applications exploiting the phone's camera.

tion 3, by a discussion of privacy and trust issues. Section 4 presents the system implementation and performance evaluation of Darwin applied to a proof-of-concept speaker recognition application. Following this, we discuss a number of other sensing applications built on Darwin and then discuss the related work in Section 5 and Section 6, respectively. Section 7 concludes with some final remarks.

2. DARWIN DESIGN

In this section, we present the detailed design of the Darwin system including the use case speaker recognition application.

2.1 Design Considerations

The design of Darwin is governed by the limited computational resources on the phone to run computationally intensive machine learning algorithms and mobility issues. In what follows, we discuss these motivating factors and the design decisions that address them.

The main goal of mobile phones is expanding way beyond just making phone calls. Compared to early mobile phones, modern smartphones are also powerful programmable platforms with up to 600MHz processors and 1GB of application memory [34]. While smartphones have increasing resources running continuous sensing applications they present a number of important challenges. These range from the design of efficient duty cycling algorithms that can maintain acceptable fidelity and time between charges to the need to push more intelligence to the phone in terms of classification algorithms without impacting the user experience (e.g., freezing the phone, slowing the UI, blocking calls). Machine learning algorithms that run on the phone to process sensor data should be implemented in an efficient lightweight manner. Darwin is designed to reduce on-the-phone computation based on a split-level computation design [29], offloading some of the work to backend servers (as discussed in Section 2) while trading off the cost for local computation and wireless communication with backend servers.

Users carrying mobile phones also presents a number of challenges for continuous sensing applications that have to operate under real-world mobility conditions. The context of the phone is challenging to sensing. Users carry phones in many different ways. Therefore, when a phone senses an event, its context (e.g., in/out of the pocket, in/out the bag, etc.) will impact the sensing and inference capability of the phone. Another challenge that mobility creates is that the same phone may sense the same type of event under different conditions (e.g., the same person speaking in a quiet office or noisy restaurant). This leads to poor inference. A group of co-located phones running the same classification algorithm and sensing the same event in time and space could compute different inference results because of the context problem and slight environmental differences, as discussed above. In essence, each phone has a different viewpoint of the same event. These real-world issues arise because sensing takes place out in the wild – not in a controlled laboratory setting – and is governed by the uncontrolled mobility of users.

Darwin exploits mobility and addresses these challenges. It uses classifier evolution to make sure the classifier of an event on a phone is robust across different environments – works indoors and outdoors for example. Extracting features from sensed events in order to train a classifier is costly for a mobile phone in terms of computation and time. Darwin allow phones to pool classification models when co-located or from backend servers. Pooling radically reduces the classification latency because a phone can immediately start to make inferences without the need to train classifiers. Different phones running the same classifier and sensing the same event are likely sensing the event differently, as discussed above. Darwin uses collaborative inference to compensate for this difference, boosting the final inference result. Darwin exploits mobility because it is designed to be opportunistic in its use of classifier evolution, pooling, and collaborative inference.

2.2 Darwin Operations

In what follows, we present a high level description of how Darwin operates: **(1)** each mobile phone builds a model[1] of the event to be sensed through a seeding phase. Over time, the original model is used to recruit new data and evolve the original model (see Figure 1(a)). The intuition behind this step is that, by incrementally recruiting new samples, the model will gather data in different environments and be more robust to environmental variations. The phone computes the feature vector[2] locally on the phone itself and sends the features to a backend server for training. This is because the feature vector computation is quicker and more energy efficient than the training phase of a machine learning algorithm such as a Gaussian Mixture Model (GMM), which is the technique we use (it takes approximately 2 hours to train a GMM with 15 seconds of audio from experiments on the Apple iPhone and N97). **(2)** When multiple mobile phones are co-located they exchange their models so that each phone has its own model as well as the co-located phones' models. Model pooling, as shown in Figure 1(b), allows phones to share their knowledge to perform a larger classification task (i.e., in the case of speaker recognition, going from recognizing the owner of the phone to recognizing all the people around in conversation). After models are

[1]A classification model is represented by a mathematical expression with parameters. For example, in the case of a Gaussian classification model, the model is identified by a Gaussian function with mean and standard deviation as the parameters. Refer to Section 2.5 for further details.

[2]A feature vector is a vector of numerical elements representing an event. For example, in the case of activity recognition applications that use the accelerometer, two feature vector elements often used are the mean and standard deviation of the accelerometer readings.

pooled from neighboring mobile phones, each phone runs the classification algorithm independently. However, each phone might have a different view of the same event – i.e., different phone sensing context. For example, one of the phones might be inside the user's pocket whereas another one might be outside, or one of the phones could be closer to the sensing event than others. **(3)** Collaborative inference exploits this diversity of different phone sensing context viewpoints to increase the overall fidelity of classification accuracy, as illustrated in Figure 1(c).

2.3 Speaker Recognition Use Case

We choose speaker recognition as our proof-of-concept application because the audio modality is generally sensitive to environment and phone sensing context and we believe the findings from this application will generalize to other classification problems such as in [27, 10] or pollution for example [20] for which the phone sensing context is important. The speaker recognition application attempts to determine the identity of a speaker by analyzing the audio stream coming from a microphone. The recognition process includes the following steps:

Silence Suppression and Voicing. The system first eliminates audio chunks that contain silence or low amplitude audio and then runs a voicing routine to remove the chunks that do not contain human voice. By focusing only on chunks that are likely to contain human speech we reduce the false-alarm rate of the classification system. The silence suppression filter works on 32 ms of audio at a time and discards portion of the audio whose root mean square (RMS) value falls below a threshold $\mathcal{T}$. The threshold $\mathcal{T}$ is determined experimentally under various conditions, for example, recording voice using the mobile phone in quiet indoor environments, on a sidewalk of a busy road, and in a restaurant. The voicing is performed by training a GMM using several hours of non-voice audio captured in various conditions (e.g., quiet environments, noisy from car traffic, etc.) and discarding the audio chunks whose likelihood falls with a +/- 5% from the non-voice likelihood. This threshold is also determined experimentally and found to be accurate for many different scenarios. More advanced techniques could be considered in future work such as the voicing scheme proposed in [11].

Feature Vector. The feature vector consists of Mel Frequency Cepstral Coefficients (MFCCs) which are proven to be effective in speech audio processing [17][16][42]. We use coefficients 2 to 20 and skip the first coefficient because it models the DC component of the audio.

Speaker Model Computation. Each speaker is modeled with a mixture of 20 gaussians (hence, a 20-component GMM). The reason we use GMM is because GMM algorithms are widely used in the speaker recognition literature [40][39]. An initial speaker model is built by collecting a short training sample – 15 seconds in our current implementation. When a user installs the application on their phone they are asked to provide a sample of voice by reading loud text displayed on the phone screen. In Darwin, the initial model evolves to capture the characteristics of the different environments where the person happens to be located.

Each speaker i's model M_i corresponds to a GMM$_i$, $\forall\ i$, i = 1..N (where N is the number of speakers), and GMM$_i$ is the model trained and evolved by P_i for speaker i[3]. A GMM$_i$ is characterized by the tuple $M_i = <\boldsymbol{\mu_i}, \boldsymbol{\Sigma_i}, \mathbf{w}_i>$, where $\boldsymbol{\mu_i}$ is the $\mathcal{P} \times \mathcal{Q}$ multi-dimensional array containing the mean values of the gaussian distribution, $\mathcal{P}$ is the number of components and $\mathcal{Q}$ is the number of dimensions of the GMM model. $\boldsymbol{\Sigma_i}$ is the $\mathcal{Q} \times \mathcal{Q} \times \mathcal{P}$ multi-dimensional covariance matrix, whereas $\mathbf{w}_i$ is the $1 \times \mathcal{P}$ array of weights for each gaussian component. For a detailed discussion of a GMM for speaker identification see [40]. The Darwin implementation uses $\mathcal{Q} = 19$ (i.e., the number of MFCCs coefficient employed) and $\mathcal{P} = 20$. We fix the number of components to 20 because the larger the number of components, where a component represents a single gaussian within the mixture, the more accurate the model is. At the same time as the number of components increases, the computing cost for the model increases. For our implementation, we experimentally verify that 20 components provide the best tradeoff between computation cost and classification accuracy.

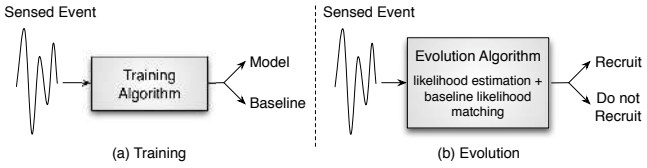


Figure 3: (a) Training and (b) Evolution steps.

Speaker Inference. In Darwin, speaker inference operates in a collaborative and distributed manner. The inference is collaborative because all the mobile phones contribute to the final results by communicating the confidence level associated with certain audio with other mobile phones in the neighborhood using the short-range radio, such as, Bluetooth or WiFi. The inference is distributed in the sense that the final inference result is derived locally on each individual mobile phone without relying on any particular external entity. We show how collaborative inference boosts the performance of the speaker recognition application.

2.4 Classifier Evolution

2.4.1 Initial Training

The initial training phase is intended to be short so that the application can be used immediately by people without requiring a prolonged initialization phase. Clearly, a short training data set implies a less accurate classification model. For this reason off-the-shelf speaker recognition applications [47][18][23] use large number of samples typically several tens of seconds of speakers' voices in order to achieve acceptable recognition accuracy. In our system, the initial speaker model is just the starting point and the model is used to recruit new training data and evolve the model on-the-go without additional user intervention. For applications other than speaker recognition, the initial model of the event to be sensed is provided by the system during the deployment phase. The evolution algorithm is designed to be applied to different sensing modalities other than audio, i.e., air quality sensing, etc. The initial training phase consists of taking a sample or seed of the sensed data and using half of the data to build the model. The remaining half for building a baseline likelihood (BL) as shown in Figure 3(a).

[3]We associate one speaker to a single phone. Whenever the speaker is using n phones we would have n different GMM models, one per phone for the same speaker.

During the initial training phase, a person is asked to talk into her phone until a voicing audio stream of 30 seconds is collected by the mobile phone. The first 15 seconds of the training set are used for training purposes and the remaining 15 seconds to set a baseline for the classifier evolution technique, as discussed in Section 4. The raw audio stream received from the microphone is first passed through the silence suppression and voicing filter; then, the training and baseline audio are both fed into the MFCCs extractor. The MFCCs^{train} computed from the training audio form the feature vector that is used to build a GMM of the speaker. The baseline audio is used to extract MFCCs^{base} and to determine the BL that is used to recruit new audio samples, as discussed in Section 2.4.2. The baseline is computed only for the model that the phone trains, which, in the case of speaker recognition, is the model of the phone owner voice.

2.4.2 Evolution

An important challenge in training classifiers is to obtain sufficient amount of labeled data for supervised training. Data labeling is tedious and expensive and real-time labeling is seldom practical since people are averse to being interrupted in order to label data. In essence, labeling data does not scale.

Therefore, machine learning algorithms running on mobile phones can not fully rely on supervised learning strategies. In contrast, a fully unsupervised learning process without any human intervention can often latch onto idiosyncrasies of the data and is not guaranteed to model the classes the user might care about. Darwin takes a different approach by using a classifier evolution algorithm on mobile phones which uses a simple semi-supervised learning strategy [54]. We show that such an approach can be effective in boosting the performance of the speaker recognition system. Darwin's classifier evolution revolves around the fact that mobile phones create different classification models for different environments if existing classification models do not fit those environments. When the speaker recognition application bootstraps, the speaker's voice is sampled in a quiet environment to build the initial model for the owner of the phone (15 seconds of voicing audio). During runtime, if the likelihood of the incoming audio stream is much lower than any of the baseline likelihoods corresponding to the different models on the phone, then a new classification model is evolved. The classifier evolution algorithm comprises the following steps: (i) obtain the likelihood for a 96 ms incoming audio chunk; (ii) compare the maximum likelihood estimator (MLE) result with BL for models already existing on the phone; (iii) recruit the new audio as part of the training data, if necessary; (iv) re-train the classification model; and finally (v) compute the new BL, which is the likelihood of the original baseline plus the newly recruited data. The classifier evolution algorithm is detailed in Algorithm 1.

Algorithm 1 Pseudo-code of classifier evolution algorithm running on node i.

```
NumEnvType ← number of environments for which a model is defined.
while there is a new incoming audio chunk do
  Compute Likelihood_new
  Compare Likelihood_new with BL_i, ∀ i, i=1..L
  if ∃ j ∈ {1..NumEnvType} s.t. Likelihood_new is within +/-3% BL_j then
    {The phone has a baseline for the environment}
    BL_comp ← BL_j
    environmentType ← j
  else
    {The phone does not have a baseline for the environment}
    BL_comp ← BL_1
    environmentType ← 1
  end if
  LikelihoodDifference ← | Likelihood_new - BL_comp |
  Add LikelihoodDifference to vecDiff
  meanDiff ← mean(vecDiff)
  stdDevDiff ← stddev(vecDiff)
  if ((stdDevDiff ≤ (previousStdDev − 5%) OR (stdDevDiff ≥ (previousStdDev + 5%))) AND (Likelihood_new ≥ BL_comp - mean(vecDiff)) then
    Recruit new training data for environmentType
    Calculate new model for environmentType
    Add data to baseline audio of environmentType model
    Calculate new BL_j
  end if
end while
```

Comparison with BL. If L is the number of classification models for different environments, the $Likelihood_{new}$ is compared with BL_i, $\forall\, i$, i=1..L. At the beginning, the phone contains only BL_1, which is derived from the initial training data set. If $Likelihood_{new}$ falls between a +/- 3% interval around one of the BL_i, then that BL_i is used as the reference BL for the environment where the speaker is situated. We determine the 3% threshold experimentally in different settings and this value is the one that best generalizes to different environments and with the best performance. Because the threshold is derived for a set of environments where people spend most of their time (e.g, quiet indoors, outdoor along a sidewalk of a busy street, and in a noisy restaurant) we are confident that the threshold would extend to other similar environments.

After having determined which model to use, the absolute value of the difference between $Likelihood_{new}$ and BL_i is recorded. The vector **vecDiff** holds these differences for each new audio chunk. The mean and standard deviation of the elements in **vecDiff** are then calculated.

Data Recruiting and Model Re-Training. New data is recruited for re-training if the following two conditions hold: *i*) the standard deviation of the elements in **vecDiff**, when the new difference is added, oscillates outside a +/-5% boundary around the standard deviation calculated before the new difference is added to **vecDiff**; and *ii)* the $Likelihood_{new}$ is greater or equal than (BL_i - mean{**vecDiff**}). The first condition ensures we are recruiting only voice data that differs from the current model and the second normalizes the likelihood with respect to the mean. As new data is added to the training set, the inference model i is recomputed and a new BL_i is calculated by adding the audio chunk to the *baseline audio* of model i. The recruiting stops

when the likelihood stabilizes inside the +/- 5% boundary because convergence is reached.

As a product of the classifier evolution algorithm, different classification models are automatically created for different environments and locally stored on the phone. We prove the effectiveness of our algorithm by running the speaker recognition application and Darwin system in three different environments, that reflect common places people find themselves. In our experiments, three models are automatically created. In our speaker recognition application each model uses 63KB of storage space and approximately 500KB of storage for the training data for each model, which includes the initial training set plus the fraction of data recruited as the model evolves. The overall storage requirement for a single speaker with three different environments is ∼1.8MB. This amount of storage is reasonable for modern mobile phones because they have several gigabytes of persistent memory and up to 1GB or more of application memory. Under more limited memory constraints the number of classification models for the same speaker and for the other speakers could be set to a maximum and the models arranged according to a circular buffer policy. As the number of models in the buffer exceeds the buffer capacity, the oldest models could be removed to accommodate new models.

2.5 Model Pooling

Model pooling is based on the simple premise of sharing classification models that have already been built and optimized by other mobile phones. It is a simple but effective way to increase classification timeliness by minimizing the inference latency. Model pooling boosts classification scalability, accuracy, and speed by providing a mechanism for fast exploitation of existing models by mobile phones rather than building classifiers from scratch themselves. Model pooling boosts classifiers scalability because models are not required to be trained for multiple events, but just for those events the mobile phone is most likely to be exposed to. For example, in the speaker recognition case the percentage of time that the phone owner's voice is captured, because of conversations with nearby people or phone calls, is greater than the percentage of time any other speaker is captured. In this case, it would be possible to accurately re-train and refine over time, using the evolution algorithm, only the phone owner's voice model rather than everybody else's voice model. Mobile phones can pool models – voice models in the case of speaker recognition – from other co-located phones for events that they do not have. These models are readily available, usable, and require no extra training steps. Model pooling does not necessarily occur between co-located phones. Models can be pooled from backend servers too. Assume a particular phone builds a sophisticated audio inference or pollution model of an area [22, 10, 20]. Phones can geo-tag models and upload them to backend servers. From this point on other phones moving into these geo-tagged areas do not have to wait to generate their own models. They simply download models from a server if available and start making inferences.

In order to formalize model pooling, let P be a mobile phone, M_p be the model derived and optimized for a certain event by P, and M_i be the classification models individually derived and optimized by N other mobile phones P_i where $i = 1 .. N$. If K of these mobile phones are co-located with P, following the pooling phase P would have its own model M_p and models M_i, where $i \in \{1 .. K\}$. Each model M_i is stored locally by P in order to be used again without the need to pool them by phone P_i, unless the model has been evolved by P_i. In that case, node P_i would announce to its neighbors that a new model has been computed and is ready to be pooled. As a result of the model pooling algorithm, all phones receive the classification models of all the other phones at the end of the model pooling phase.

After node P has pooled a classification model $M_i = <\boldsymbol{\mu_i}, \boldsymbol{\Sigma_i}, \mathbf{w}_i>$ from node P_i, node P will replace P_i's classification model M_i only if P_i announces a new classification model M_i^{new}. Node P_i determines that it is time to announce the new classification model M_i^{new} when the evolution of the model is complete.

2.6 Collaborative Inference

Collaborative inference is designed to boost the classification performance by taking into account the classification results of multiple phones "observing" the same event instead of relying on only individual classification results. The idea behind collaborative inference is as follows: given that an event could be sensed by multiple, distributed but co-located mobile phones, Darwin leverages the classification results of individual phones to achieve better accuracy. The hypothesis is that we can take advantage of the multiple sensing resources around an event and *exploit their spatial distribution and context diversity.* In the case of speaker recognition, when a person is talking with a group of people, the phones of the other people would pick up the speaker's voice with different characteristics. Some phones could be in pockets, others on a clip belt or closer to a source of event than the other phones. In this case, if the application relies only on each individual phone's classification, the speaker classification accuracy would be poor when the phone sensing context is not good (e.g., in pocket or near a noise source) in relation to the specific event. The same reasoning holds for pollution measurements using mobile phones.

Following the pooling phase, each phone P_i contains the other phones classification models M_i, where $i \in \{1 .. K\}$, and K is the number of neighbors. Each phone P_i runs the inference algorithm in parallel using the common set of models. The fact that the event is classified by multiple phones using common classification models and that these phones may be exposed to different contexts (some of which are better than others) can lead to higher accuracy when the results are combined. The collaborative inference phase breaks down into three distinct steps: *i)* local inference operated by each individual phone; *ii)* propagation of the result of the local inference to the neighboring phones; and *iii)* final inference based on the neighboring mobile phones local inference results.

2.6.1 Local Inference

During the local inference phase each node individually operates inference on the sensed event using its own inference model of the event and the inference models pooled from other phones. The goal is to locally derive the confidence level of the inferred event in order to be able to communicate this confidence level to neighboring phones. By having the confidence level of each phone sampling the event, all the phones are able to run the collaborative inference step to compute the final inference result. In order to operate collaborative inference, the phones must be

time-synchronized because they need to perform inference on the same event at the same time. We rely on the fact that mobile phones support time synchronization through the cellular infrastructure. We measure a time synchronization error between four iPhones synchronized through the cellular network of 500 ms. If the error is larger than 500 ms we use a loose synchronization approach. One of the phones (randomly picked) sends a broadcast message which, when received by all the neighbors at the same time, triggers the sampling phase. After having received this broadcast message phones are synchronized.

Following the audio signal sampling and filtering, the stream is divided in 96 ms long chunks[4]. MFCCs are extracted from each chunk. At this point, a maximum likelihood estimation algorithm is run in order to verify which of the models best fits the audio chunk. To avoid having the maximum likelihood estimator running through too many pooled models, which could potentially be costly in terms of computation for a mobile phone, the estimator only uses the models of the phones detected in the neighborhood. Neighbor detection is performed using short-range radio technology, such as, Bluetooth and WiFi.

In order to increase the classification accuracy we feed the maximum likelihood estimation result into the *windowing* block. After the windowing step the local classification result is derived. If the maximum likelihood estimator returns that the $i-$th audio chunk belongs to event E, the event E is deemed to be true if and only if the following two conditions hold: (i) event E, i.e., a certain speaker speaking, is detected at least once in a window comprising the previous two audio chunks; and (ii) the classification accuracy confidence is at least 50% larger than the confidence for any other events. The reason for the first condition is to guarantee that a voice sample is included – since audio chunks are 96 ms long, if a speaker is detected at least twice in a window of time of duration 96 ms $\times$ 3, then we have better confidence that we are really capturing that speaker. A larger window size would increase the classification latency and after experimentation we determine that a window size of 3 best suits our system, which requires near-real time classification results. The reason for the second condition, having determined the 50% threshold experimentally, is to dampen the impact of false positives. The pseudo-code of the local inference algorithms is shown in Algorithm 2. The locally inferred speaker ID is associated with the classification model that best fits the audio chunk following the windowing policy. Because a phone computes the inference confidence for each of the K models as reported in Algorithm 2, the result of the local inference takes the form of the following vector $\mathbf{LI}^j_{1..s} = \{confidenceSpeaker_1, confidenceSpeaker_2, .. , confidenceSpeaker_k\}$, where the subscript notation $1..s$ indicates that the vector contains the inference results for speakers 1 to s and the superscript index j indicates that the vector is generated by node j. Consider for example, that three nodes (N_1, N_2, and N_3) are co-located and running the speaker recognition application. If S_1, S_2, and S_3 are the speakers associated with nodes N_1, N_2, and N_3 respectively, and assuming S_1 is actually speaking, then an output for the **LI** vectors could, for example, be: $\mathbf{LI}^1_{1,2,3} = \{0.65, 0.15, 0.2\}$, $\mathbf{LI}^2_{1,2,3} = \{0.4, 0.5, 0.1\}$, and $\mathbf{LI}^3_{1,2,3} = \{0.7, 0.2, 0.1\}$. The reason of N_2 expressing lower confidence about speaker 1 could be caused by the fact that mobile phone N_2 may not have a good sensing context.

[4]We use 96 ms for each audio chunks to make it a multiple of the MFCC binning size of 32 ms. This multiple makes the duration of the audio chunk small enough to maximize the likelihood of capturing just one speaker at a time.

Algorithm 2 Pseudo-code of the local inference algorithm running on node i using K pooled models. The number K is determined by the number of detected neighbors. The locally inferred speaker ID corresponds to the index of the model best fitted by the audio chunk following the windowing policy.

while there is a new incoming audio chunk **do**

{Go through all the classification models:}

for $j = 1$ to K **do**
arrayLE[j] ← likelihood estimation for audio chunk
end for

Retrieve the max likelihood from arrayLE and the associated index indexMA.
Retrieve the second larger confidence from arrayLE and the associated indexSMA.

{Windowing policy:}

if $(arrayLE[indexMA] - arrayLE[indexSMA] \geq 0.5)$ AND $((indexMA == previousMAX)$ OR $(indexMA == previousPreviousMAX))$ **then**
locallyInferredSpeaker ← indexMA
previousPreviousMAX ← previousMAX
previousMAX ← indexMA
end if

end while

2.6.2 Local Inference Propagation

Following the local inference step, each mobile phone i broadcasts the result of its local inference (i.e., the vector $\mathbf{LI}^i$) in order to give neighboring phones the ability to move to the final step, i.e., the final inference. A time stamp is associated with vectors $\mathbf{LI}^i$ in order to align local inference results in time. A weight is also associated with vectors $\mathbf{LI}^i$. The weight is an indication of the quality of the local inference. For example, assuming the mobile phone determines its context (e.g., in or out of the pocket) and is running an audio or pollution sensing application [20], then the value of the weight is small or large when the phone is in or out of the pocket, respectively. Therefore, a $\mathbf{LI}^i$ vector with a small weight indicates that the mobile phone i is not in a suitable sensing context and has less influence on the final inference phase.

The phones broadcast their local inference results at the rate the local inference is computed. The local inference is computed right after an audio sample is collected. Consequently, if the phone polls the microphone every A number of seconds, the local inference is computed and broadcasted every A seconds as well. For example, a local inference broadcast rate of 1Hz implies receiving the local inference results, hence, computing the final inference, every second. Clearly, the higher the broadcast rate the faster the final inference (see Section 2.6.3). We do not propose any specific local inference broadcast rate, since this depends on the application requirements. Rather, we show the cost of transmitting

a single confidence level from which the total cost for each specific transmission rate is derived (see Section 4.4).

2.6.3 Final Inference

The goal of the final inference phase, which follows the local inference propagation, is to compensate event misclassification errors of each individual mobile phone achieving better classification confidence and accuracy of sensed events. Mobility and context affect the local inference result, sometimes such that an event could be misclassified. For the final inference, Darwin combines the local inference results derived by individual mobile phones that are spatially distributed in the area of the target event. The technique used to perform the final inference is to find the inferred speaker (IF) that maximizes the joint probability as in Equation 1.

$$IF = \underset{\mathbf{j}, j \in \{1..S\}}{\arg\max} \{Prob(s_j^1), Prob(s_j^2), .., Prob(s_j^K)\} \quad (1)$$

where K is the number of co-located phones that have been contributing with their local inference, S is the number of speakers, and $Prob(s_j^i)$ is the probability that speaker j is detected by phone i. This operation is facilitated by the fact that the property of independence between the different $\mathbf{LI}_{1..S}^j$ vectors, $\forall j$, $j = 1..K$, is verified since each node (i.e., phone) N_i, $i = 1..K$, performs independent event sensing. Given the property of independence, we can re-write Equation 1 as:

$$IF = \underset{\mathbf{j}, j \in \{1..S\}}{\arg\max} \{\prod_{i=1}^{K} \mathbf{LI}_1^i, \prod_{i=1}^{K} \mathbf{LI}_2^i, .., \prod_{i=1}^{K} \mathbf{LI}_S^i\} \quad (2)$$

Since each node N_i has the vectors $\mathbf{LI}_{1..S}^k$ (where k $\in$ $\{1,..,K\}$) after the local inference propagation phase, each node N_i can compute Equation 2 to produce the final inference. In order to assure that Equation 2 is calculated using the local inference results of the same event, Equation 2 is computed only over the local inference results that are aligned in time. For each computation of Equation 2 only the local inference results which differ in time for at most δ ms are considered. In our implementation, we set δ = 150 ms, which we determine experimentally to be the best value. The larger δ, the bigger is the likelihood that the local inference results refer to different events. The smaller δ, the higher is the likelihood to capture just one event, but the closer we get to the phones time synchronization error. Because time stamps are associated with **LI** vectors (see Section 2.6.2) it is possible for the phone to determine if an event is sampled at the same time.

In order to provide a more robust system against false positives, a windowing approach is adopted where a speaker is deemed to be speaking only if they are detected for at least one more time in the past two iterations of Equations 2. This policy, similar to the windowing policy introduced for the local inference in Section 2.6.1, provides experimentally the best tradeoff between classification delay and classification accuracy.

3. PRIVACY AND TRUST

Security, privacy, and trust raise considerable challenges in the area of mobile phone sensing. While we do not present solutions to these challenges, those solutions are critical to the success of the research discussed in this paper. Darwin incorporates a number of design decisions that are steps towards considering these challenges. First, the raw sensor data never leaves the mobile phone nor is it stored on the phone - we only store models and features computed from the raw sensor data. However, features and models themselves are sensitive data that needs to be considered appropriately and therefore protected. In the case of the speaker recognition application, the content of a conversation is never disclosed, nor is any raw audio data ever communicated between phones. The data exchanged between phones consists of classification confidence values and event models. Next, mobile phone users always have the ability to opt in or out of Darwin, hence, no model pooling and collaborative inference would take place unless the users make such a determination.

To meet privacy, security, and trust requirements Darwin phones should: i) run on trusted devices; ii) subscribe to a trusted system; and iii) run a trusted application that is either pre-installed on the phone or downloaded from a trusted third party (e.g., Apple App Store, Nokia Ovi Store, or Android Market). Any phone-to-phone interaction (e.g., pooling and collaborative inference) should be regulated by off-the-shelf authentication and authorization mechanisms that prevent the injection of malicious code or intentionally distorted inference results from adversaries.

4. SYSTEM PERFORMANCE

In what follows, we first discuss the implementation of Darwin on the Nokia N97 and Apple iPhone and then present the detailed evaluation results of the Darwin system supporting the speaker recognition application.

4.1 Phone Implementation

Darwin is implemented on the Nokia N97 using C++, Kiss FFT [4] for fast fourier transform (FFT) calculations, and QT [7], which is a wrapper around C++ for the graphical user interface. On the Apple iPhone we use C++ and the FFTW fast fourier transform library [3]. The necessary algorithms, i.e., GMM training, the probability density function, and MFCC extraction are ported to the N97 and iPhone from existing Matlab code that we verified to work correctly. We plan to make this toolkit available in the future as an open source project. The availability of this toolkit on a phone is a considerable resource for building more powerful classifiers. The backend server responsible to run the model training and re-training for evolution is implemented on a Unix machine using C and standard socket programming for communications. A UDP multicast client is implemented to allow local inference results propagation whereas an ad-hoc lightweight reliable transport protocol has been built to send feature vectors to the backend for model training, to send trained models to the phones, and to exchange models during the pooling phase. Because we target heterogeneous scenarios, an audio sampling rate of 8KHz is used in order to run Darwin on the iPhone 3G, which can drive the microphone up to 48KHz sampling, the iPhone 2G and the Nokia N97, which only support 8KHz audio sampling rate.

4.2 Experimental Results

We evaluate the Darwin system using a mixture of five N97 and iPhones used by eight people over a period of two weeks generating several hours of recorded audio containing

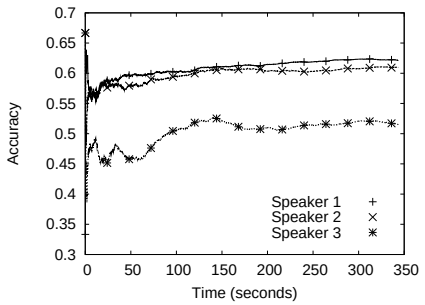


Figure 4: Accuracy, without evolution, for three speakers when walking along a busy road without classifier evolution and having trained the classification models for indoors only.

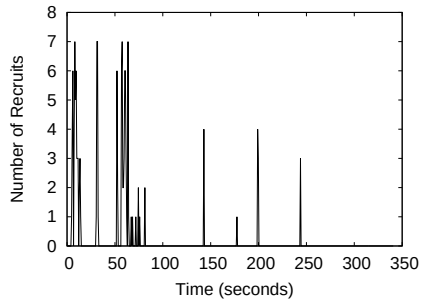


Figure 5: Evolution sessions count over time in the indoor scenario for speaker 8.

speech portions. In order to evaluate the system against ground truth data, the audio data is manually labeled by extracting the voicing chunks of each of the eight speakers. The audio is recorded in different locations under differing conditions such as a quiet indoor environment, walking on a sidewalk along a busy and noisy street, and in a noisy restaurant. This provides a good basis to validate Darwin under very different operating conditions. We only present a subset of the results from our experiment due to space limitations.

4.2.1 The Need for Classifier Evolution

We conduct a simple experiment to show the need for classifier evolution on mobile phones. Three people walk along a sidewalk of a busy road and engage in conversation. The speaker recognition application without the Darwin components runs on each of the phones carried by the people; that is, no classifier evolution, model pooling, and collaborative inference algorithms are enabled for the experiment. The voice classifier for each person is trained in a quiet indoor environment. We quantify the performance of a classification model trained indoors when operating in a noisy outdoor setting. Figure 4 shows the classification accuracy [53] for mobile phones 1, 2, and 3 for speaker 1, 2, and 3, respectively. It is evident from the plot that the accuracy is quite low because the maximum speaker classification accuracy for speaker 1 and 2 is 63% and 61%, respectively, and only 52% for speaker 3. The poor performance is because the classification model trained indoor performs poorly outdoors. This highlights the challenge presented when sensing in different environments. Building audio filters capable of separating

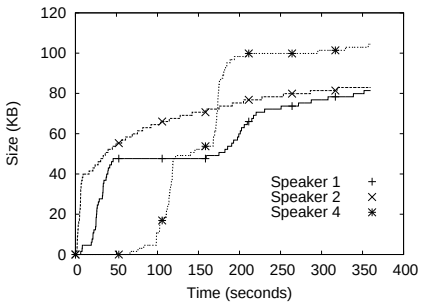


Figure 6: Size of the data set recruited during the evolution phase in the restaurant scenario.

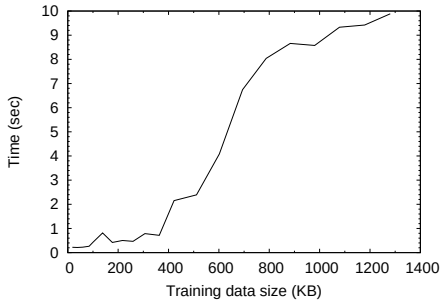


Figure 7: Server training computation time as a function of the training data size. The server has a 2.4GHz cpu and 4GB of RAM.

voice from other types of noise is challenging and would not likely scale given the large pool of possible sounds that may be encountered by a mobile phone on the street or in any other environments. This result motivates the need for designing efficient classifier evolution techniques that operate transparently to the person carrying the phone in order to evolve classification models according to the scenarios where people find themselves.

Let us take a look now at the evolution algorithm performance. Figure 5 shows the distribution for the duration of the experiment of the number of audio chunks recruited by the classifier evolution algorithm for speaker 8 in the indoor case. Similar results are observed for other speakers not shown in the results. As expected, a larger number of chunks are recruited during the first phase, when the application is run after the initial training, than towards the end, when the model has already been refined and little or no model evolution is required.

Figure 8 shows the accuracy improvement as the amount of data sent from the phone to the backend for re-training grows. This result refers to a different outdoor environment than the one in Figure 4. This second outdoor scenario is noisier than the first one, causing the initial accuracy before evolution to be lower than the one reported in Figure 4. Clearly, the larger the training set capturing the characteristics of the new environment the better the performance. Figure 6 shows the amount of audio data recruited by the evolution algorithm for the duration of the restaurant experiment. It is interesting to see that after a few minutes of conversations the models of the three speakers diminish their training data recruitment and model evolution eventually stops (as happening for speaker 1 model). This confirms

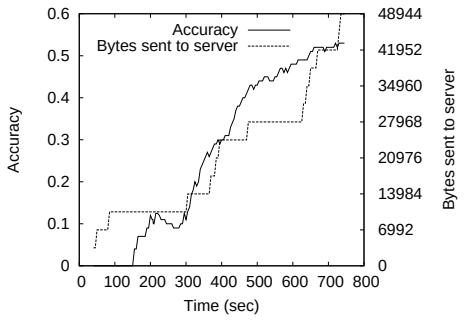


Figure 8: Classifier accuracy and the amount of needed training data in an outdoor scenario.

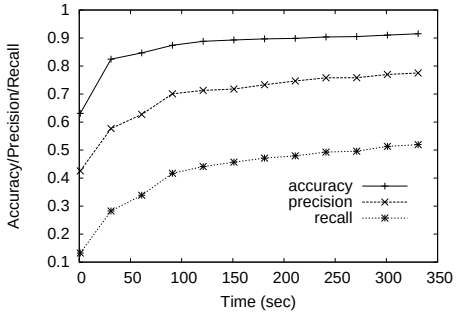


Figure 9: Mean recall, precision, and accuracy in indoor quiet environment with collaborative inference for the eight speakers.

the behavior observed for Darwin in the quiet indoor environment. The only difference is in the amount of recruited data between the restaurant and quiet scenario. In fact, in the former case the amount of data recruited ranges between 78KB and 100KB, whereas in the indoor case it is between 16KB and 70KB. This is in line with the fact that less data is needed to evolve an indoor model than a "restaurant" model (i.e., a very different environment from a quiet place) since initial training is performed in a quiet place. Figure 7 reports the server training algorithm running time as the training data set grows. The measurement refers to the outdoor case and levels off when the accuracy (shown in Figure 8) reaches the maximum. Right before the end of evolution the server takes about 10 seconds to train the GMM model using a 1.3MB data set.

4.2.2 *Experimental Scenario One: Quiet Indoor Environment*

We first show the performance of the speaker recognition application analyzing data collected from 5 different phones concurrently running the Darwin system in a meeting setting in an office environment where 8 people are involved in a conversation. The phones are located at different distances from people in the meeting, some on the table and some in people's pockets. The aim of the experiment is to study the impact of different phone sensing context showing how a low classification accuracy due to adverse context is compensated by Darwin. In Figure 11 we show the classification precision [53] for speaker 8 calculated by speaker 8's phone using the basic speaker recognition application when speaker 8 is talking. Figure 11 shows that the precision of

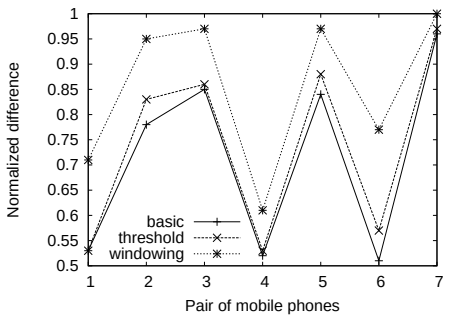


Figure 10: Normalized true positive-false positive difference between speaker 8 and all the other speakers when speaker 8 is speaking. The closer the normalized difference to 1, the larger is the true positives compared to false positives.

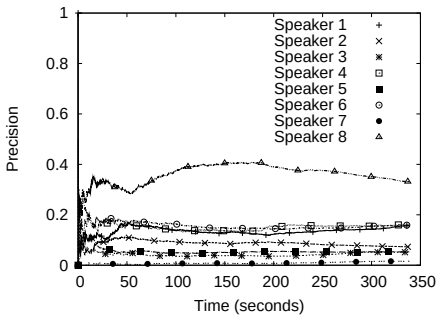


Figure 11: Precision for speaker 8 basic inference when speaker 8 is speaking in an indoor quiet setting.

the basic speaker recognition application is below 50%. This result indicates that speaker recognition using an individual mobile phone is challenging. The reason is that the phone could have poor sensing context for example in the pocket (as for part of the conversation of speaker 8) or affected by other factors such as noise mixed with voice.

Figure 10 shows the benefit of applying the 50% thresholding technique and windowing policy (described in Section 2.6) to the basic speaker recognition classification algorithm. The 50% thresholding technique makes sure that there is a 50% difference between the confidence of the inferred speaker and every other speaker, whereas the windowing policy reduces the effect of false positives. Figure 10 shows the difference between the true positive and false positives counts normalized by the true positive count for speaker 8 speaking. The closer the normalized difference to 1, the larger is the number of true positives versus the number of false positives. Figure 10 shows the combined positive effect of the 50% thresholding and windowing techniques, which makes the normalized difference larger compared to the basic technique. This is a first step toward the final inference, which is part of the collaborative inference phase of Darwin; however, it is not enough to achieve higher classification accuracy. In fact, when the Darwin system is activated a further performance boost is registered. Figure 9 shows results for the mean recall [53], precision, and accuracy results for the eight speakers classification when Darwin is running. It is evident from the plot that Darwin boosts

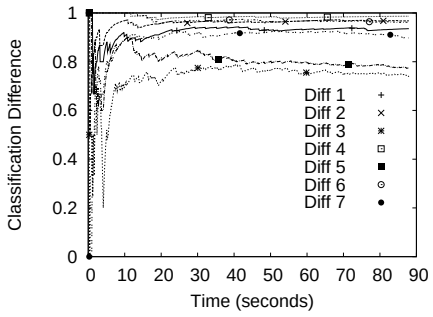


Figure 12: Classification difference between speaker 8 and the other speakers without Darwin.

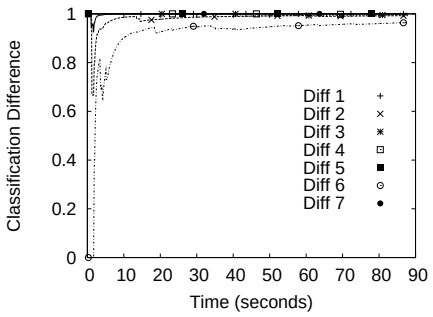
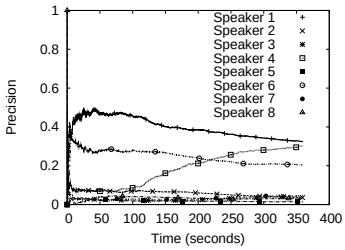
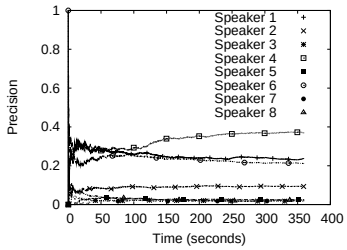


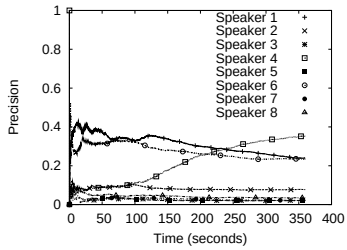
Figure 13: Classification difference between speaker 8 and the other speakers with Darwin.



(a) Precision for speaker 4 calculated by the speaker's mobile phone without collaborative inference.



(b) Precision for speaker 4 calculated by mobile phone 1 without collaborative inference.



(c) Precision for speaker 4 calculated by mobile phone 2 without collaborative inference.

Figure 14: Precision for speaker 4 on different phones in a noisy restaurant environment without collaborative inference.

the classification performance of the speakers identification. The boost is due to collaborative inference, which leverages the other co-located mobile phones in order to achieve better classification accuracy.

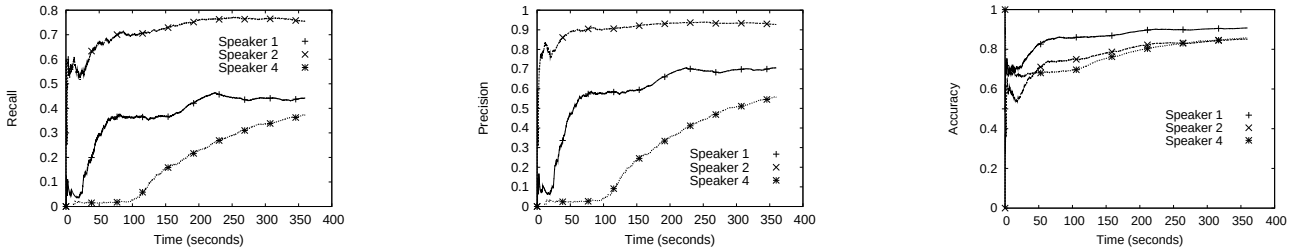
The other interesting result from Figure 9 is the positive impact of Darwin's classifier evolution. The accuracy, precision, and recall increase over time at the beginning of the experiment as new audio is recruited and the classification models re-trained taking into account the new audio. The performance for accuracy, precision, and recall levels off at the point where the classifier evolution algorithms does not benefit from more data, as discussed in Section 2.4.

The benefit of model evolution, model pooling, and collaborative inference can be also seen in the results shown in Figures 12 and 13. If we indicate with TP and FP, respectively, the true and false positives when speaker 8 is speaking, the y-axis of the plots reports the quantity (TP-FP)/TP over time. A FP count is maintained for each speaker, thus in Figures 12 and 13 eight curves are shown. In one case Darwin is disabled (Figure 12), in the other case Darwin is enabled (Figure 13). When (TP-FP)/TP is close to 1 it means that the number of true positives dominates the number of false positives. In contrast, if (TP-FP)/TP is close to 0 we have that the number of false positives approximates the number of true positives. We can see that the difference between speaker 8's true positives and any other speakers' false positives is larger when Darwin in running (as shown in Figure 13) than when it is not (see Figure 12). This is another indication of how Darwin improves the classification result for a given speaker.

4.2.3 *Experimental Scenario Two: Noisy Indoor Restaurant*

In the next experimental scenario, we evaluate Darwin when the speaker recognition application is running on five phones while five people are having a meal in a noisy restaurant. This contrasts the first scenario of a quiet indoor setting. Three out of five people are engaged in conversation. Two of the five phones are placed on the table, the other phones are in the pants pockets for the entire duration of the experiment. In Figure 14, we show the classification precision of speaker 4 from three of the mobile phones located around the table people are sitting at; note, we observe similar trends for the other phones. Figure 14(a) is the precision computed by speaker 4's phone, which is the closest phone to speaker 4 (it is carried by speaker 4) for when speaker 4 is talking. The reason we select speaker 4 for the evaluation is that speaker 4 is the closest person to a nearby table where another group of people is having a loud conversation. Here we show the benefit of the Darwin system for the phone of a speaker who is positioned in a non optimal context, i.e., close to a noise source.

Figures 14(b) and 14(c) refer to the precision calculated by two other phones, which we call phone 1 and 2, located at the opposite side of the table where speaker 4 is sitting. Figures 14(b) and 14(c) show on average higher classification precision on phones 1 and 2 when speaker 4 is talking than when the classification is performed by speaker 4's phone reported in Figure 14(a). This is because phones 1 and 2 are more distant from the source of noise and consequently they are able to capture higher quality speaker



(a) Recall. (b) Precision. (c) Accuracy.

Figure 15: Recall, precision, and accuracy in a noisy restaurant with Darwin for three speakers.

4's voice audio than speaker 4's phone itself. The important point is that Darwin boosts the classification performance of speaker 4 by leveraging other surrounding phones experiencing better sensing contexts; this is shown in Figures 15(a), 15(b), and 15(c) which report, respectively, the recall, precision, and accuracy of the three speakers in the restaurant experiment, including speaker 4, over time. From Figure 15(c) it can be seen that speaker 4's accuracy is low (∼65 %) at the beginning of the experiment but starts increasing as the experiment proceeds. The classifier evolves and the surrounding mobile phones participate in collaborative inference. At the end of the experiment, speaker 4's classification accuracy reaches ∼80%. The fact that speaker 4's recall and precision present low values for the duration of the experiment (as shown in Figures 15(a), 15(b), respectively) confirms that speaker 4 voice is impacted most of the time by loud conversation from the next-table.

4.2.4 *Experimental Scenario Three: Walking Outdoors Along a Sidewalk in a Town*

The final scenario we study is an outdoor environment where five people walk along a sidewalk and three of them are talking. This contrasts the first two scenarios. In this experiment, five people carry phones either clipped to their belt or inside their pockets. As in the restaurant scenario, the amount of audio data recruited by Darwin to evolve the classifier is larger than the indoor evolution case and ranges between 90KB and 110KB of data. The performance boost using the Darwin system can be observed in Figure 16 where the speaker recognition classification accuracy increases to 80-90%. The greatest improvement is observed by speaker 1 whose phone is clipped to their belt. This mobile phone is exposed to environmental conditions such as wind and passing cars making the audio data noisy and voice hard to pick up. However, we note that some of the other phones experience better sensing context and by relying on these phones Darwin boosts the final classification accuracy for speaker 1.

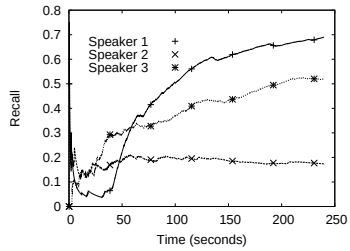
4.3 Impact of the Number of Mobile Phones

In this experiment, we study how the Darwin system's performance changes as the number of mobile phone participating in model pooling and collaborative inference varies. This experiment is also conducted in the same noisy restaurant discussed in scenario two, which represents a challenging sensing environment. The experiment consists of three people speaking and five phones carried by five people positioned around the table. Some phones are placed on the table and others remain in speaker's pockets. The experimental scenario starts with only two of the five phones running the Darwin system. More nodes are subsequently added up to a maximum of five – all phones run the Darwin system and are randomly positioned around the table. The classification accuracy for each of the three speakers as a function of the number of phones running Darwin is shown in Figure 17. As expected, the larger the number of co-located mobile phones running Darwin, the better the inference accuracy. The performance gain using collaborative inference grows with the number of phones according to the algorithm discussed in Section 2.6 and in particular to Equation 2. While two phones do not have sufficient "spatial" diversity to experience gain from the Darwin system, the classification accuracy is boosted when the three remaining phones are added.

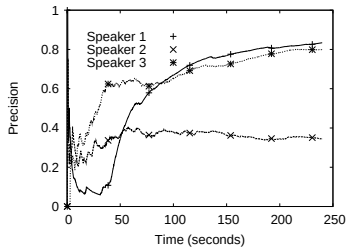
Speaker 3 experiences low accuracy due to the proximity of their phone to another group of people involved in a loud conversation. This is perceived as noise by the speaker identification classifier and negatively impacts speaker 3's voice classification more than speaker 1 and 2. Speaker 2 experiences low accuracy with three phones running Darwin due to speaker 2's voice characteristics. Speaker 2's classification model poorly classifies speaker 2's voice when operating individually. This could be due to the fact that the initial training audio is not recorded correctly or that the 20-component 19-dimensional GMM for speaker 2 does not properly model speaker 2's voice. In this case, a larger number of nodes is needed to perform accurate speaker 2 classification. The Darwin system compensates not only errors due to different sensing context but also for poor event classification modeling. This is possible because multiple phones co-operate to generate more accurate inference results. The confirmation that speaker 2's model is not accurate comes from the fact that speaker 2's recognition with 3 phones performs poorly in comparison with speaker 1 and 3 in a quiet setting (see Figure 18), which is where the classifier should perform the best given the initial indoor training stage.

We also determine that the reason for better inference accuracy with 3 phones in the restaurant experiment for speaker 3 is that the other two phones are closer to speaker 3 than they are in the quiet indoor case. This offers better audio signal quality for the collaborative inference step.

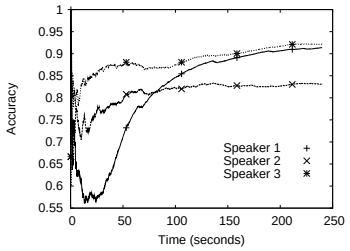
In summary, the Darwin system boosts the classification accuracy when the sensing environment or context adversely impacts quality of inference, when the individual classification model yields poor accuracy given a person's voice characteristics (as in the case of speaker 2 for our experiments), and when sensors or microphones have different hardware characteristics [39].



(a) Recall.



(b) Precision.



(c) Accuracy.

Figure 16: Recall, precision, and accuracy for three speakers walking on a sidewalk along a busy road.

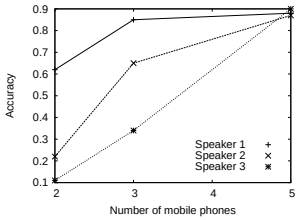


Figure 17: Accuracy in a noisy restaurant when an increasing number of phones participate to Darwin.

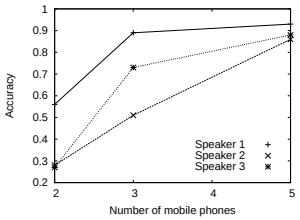


Figure 18: Accuracy in a quiet indoor setting when an increasing number of phones participate to Darwin.

4.4 Time and Energy Measurements

When proposing a complex but powerful classification architecture such as Darwin the natural question is: how does this system impact the performance of everyday mobile phones? While we have completed a first implementation of Darwin on the Nokia N97 and Apple iPhone we recognize that there are challenges and future work to be done. In what follows, we present some time and energy measurements for Darwin running on the N97 (similar performance is observed for the iPhone). We believe that smart duty cycling is also a future part of our work which would improve the energy results presented in this section. Averaged baseline measurement are taken before each measurement in order to have a baseline reading, which we subtract from each measurement. The measurements are performed using the Nokia Energy Profiler tool [5] and repeated five times. The mean values are reported. The running time of each Darwin component for 1 second of audio sampling is reported in Table 1. The most computationally intensive routines are the local inference (which involves the probability density function calculation for eight speakers) and receiving the model from the server. Figure 19 shows the power, CPU load, and memory measurements on the N97 when running the Darwin components. It can be seen that the local inference step takes the largest amount of power (see Figure 19(a)) and CPU load (see Figure 19(b)). Substantial memory usage is measured for MFCC extraction compared to the other components (see segment (C) of Figure 19(c)). This suggests that the MFCC extractor implementation requires optimization. Receiving a new model from the server and broadcasting it to neighbors during pooling also causes more power drain than the other routines. However, evolution and pooling are operations that occur rarely (i.e., evolution only once for a certain environment and pooling only once to send a model to neighbors), consequently, pooling and evolution do not heavily account for resource usage and power consumption. Routines that instead occur periodically, such as audio sampling, voicing, MFCC extraction, etc., require less power each time they are active.

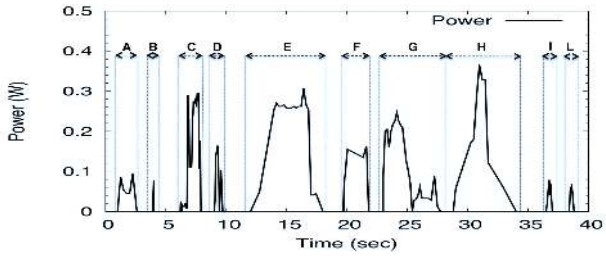
Table 1: Average running time for processing 1 sec audio clip, sending, and transmitting data.

Routine	Running Time (s)
Silence suppression	0.003
Voicing	0.565
MFCC extraction	1.4
Local inference	3.67
TX MFCC to server	0.862
RX model from server	4.7
TX model to neighbors	1.91
TX local inference	0.127
RX local inference	0.09

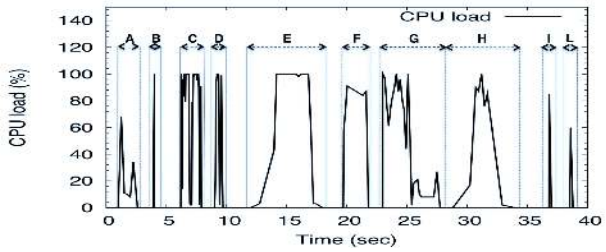
Finally, Figure 20 shows the measured battery lifetime and the inference responsiveness (defined as the inverse of inference delay) as a function of the audio sampling interval and collecting 1 second of audio each time the microphone is polled. We obtain the shortest battery lifetime ($\sim$27 hours) for a periodic sampling interval of 10 seconds (this sampling interval guarantees the highest inference responsiveness). However, if smart duty-cycling techniques are adopted [48], the phone could operate in a low sensing duty-cycle mode, e.g., with a sampling rate of 60 seconds, when Darwin is not running. As an event is detected, such as voice in case of speaker recognition, Darwin could become active in a high duty-cycle mode, e.g., using a 10 second sensor polling rate, for the duration of the event. As the event disappears the phone could go back to low duty-cycle mode and Darwin would stop working. This would guarantee high application responsiveness while maintaining several hours of battery duration. More detailed analysis of resource consumption and the development of low-energy duty cycling for Darwin are important future work. We believe however that new duty cycling techniques discussed in the literature for mobile phones [48, 32] could boost the phone's battery lifetime of Darwin phones.

5. DEMO APPLICATIONS

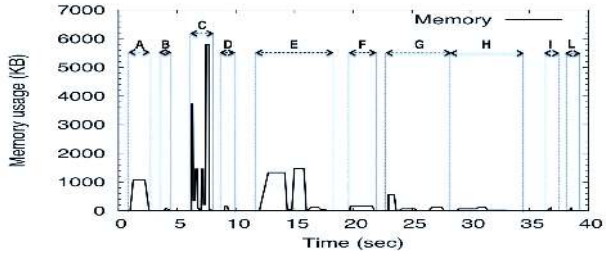
Darwin can be used by other emerging mobile phone sensing applications in addition to the speaker recognition application discussed in this paper. In what follows, we discuss how a number of demo applications that use different sensing



(a) Power.



(b) CPU load.



(c) Memory.

Figure 19: Power, CPU load, and memory usage of the N97 when running Darwin components. The components have been made run sequentially and the operations have been segmented as reported in the following labels: (A) One sec audio sampling; (B) Silence suppression; (C) MFCC extraction; (D) Voicing; (E) Local inference; (F) MFCC transmission to the server; (G) Model reception from the server; (H) Model transmission to neighbors; (I) Local inference broadcast; (L) Local inference reception.

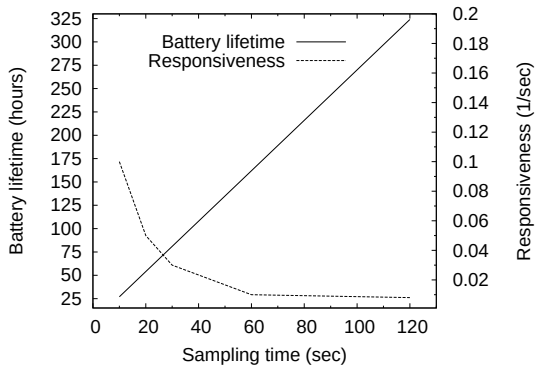


Figure 20: Battery lifetime Vs inference responsiveness.

Figure 21: Virtual Square, an augmented reality application on the N97 with the support of Darwin.

modalities can be supported by Darwin. We discuss three different demo applications.

5.1 Virtual Square Application

Darwin could support applications in the social sphere setting, where speaker identification, for example, could be used to augment the context of a person and their buddies and make the buddy proximity detection more accurate. Similarly, it could be used for reliable detection of meetings. A pure Bluetooth or WiFi based proximity detection system might not work accurately considering the large amount of devices a Bluetooth or WiFi scan could potentially return in the area where the person is. We have developed Virtual Square, an application that exploits augmented reality to present a person's sensing status information [29] including whom the person is in proximity/chatting with at a certain moment and location. Our Virtual Square prototype is built for the Nokia N97 writing a combination of QT code, for the body of the program, and Symbian routines to access low level functionalities for the collection of magnetometer readings from the onboard magnetometer. GPS and magnetometer sensors are used to geo-tag the person's sensor context which is stored on a server and is retrievable by the buddies who subscribe to the service. Users have full control of the application privacy settings and can opt-out from disclosing who they are chatting with at any time. A screenshot of the application is reported in Figure 21. It is clear how Virtual Square, by simply pointing the phone as when taking a picture and moving it around, is a powerful means to see "through walls" and gather information about people and places in the very intuitive way of an augmented reality interface. The old concept of square as a place where people gather to chat and meet now becomes virtual, being enabled by the smartphones sensors which allow to characterize people's microcosmos.

5.2 Place Discovery Application

Darwin could support applications using different sensing modalities; for example, place discovery applications based on radio frequency (RF) activity from WiFi access points [22]. Darwin can be integrated with such an application in the following way:

- **Initial training and evolution:** the classification model in this case is the RF signature profile that characterizes a certain place. The dynamics of the radio characteristics and the fact that access points are being added or removed in certain areas make the RF profile time varying. An RF profile could be initially built by a mobile phone and then

evolved as the mobile phone visits the same place multiple times in the future.

- **Pooling:** when a mobile phone visits an area for which it does not have an RF profile it has two options: either build a profile, which requires time and introduces delay in inference, or, pool a profile for the area from a nearby mobile phone or backend server. Building an RF profile could take more time than the duration of the stay in the area, which means that the place might not be discovered. By pooling, the profile is immediately ready to be used in the inference phase.

- **Collaborative inference:** if multiple mobile phones are co-located in the area they can co-operate to perform a more accurate place inference. Given that sensed RF activity could be slightly different from phone to phone, collaborative inference could be used to determine what is the most likely discovered place by, for example, selecting the place that is reported with highest probability by each of the mobile phones.

5.3 Friend Tagging Application

The idea of this application is to exploit face recognition to tag friends on pictures. Namely, the application automatically associates a name to a person in the picture if the person is recognized. Darwin could improve the application in the following manner:

- **Initial training and evolution:** the initial training starts on each user's mobile phone. The mobile phone derives a model for the person's face through a training picture. Following this initial training seed, the face model for a person can evolve over time. For example, a person's face is often captured by the phone's camera (e.g., when using video conferencing) allowing the model to be refined under different conditions (varying light conditions, on the move, etc).

- **Pooling:** when friends get together their phones pool each other's face models. Person A's phone does not have to derive a face model for person B. It pools it directly from person B's phone.

- **Collaborative inference:** face detection can now be run in parallel to tag friends when taking group pictures. Imagine a group of co-located friends taking pictures of each other. Each picture could have different people and the lighting and angle of each shot could vary considerably. Co-located phones individually run their face classification algorithm and then exchange information to refine the final inference; for example, tagging the people that the local inferences returned with highest confidence.

6. RELATED WORK

Work on applications and systems for sensing enabled mobile phones is growing in importance [31, 9, 35, 8, 19, 21, 13, 27, 10, 30, 29]. Most of the work in the literature, however, propose local sensing operations running on individual devices and do not exploit in-field mobile phones interactions. An exception to this is the work in [19], which considers context driven sampling and calibration techniques for mobile sensor networks.

Sensor node co-operation is studied mainly in the context of static sensor networks where fusion [24, 38, 49] and aggregation [52, 45, 33] techniques are applied. The benefit of sensor nodes cooperation in the context of object tracking using distributed Kalman Filters is discussed in [36, 37]. In [24] the authors propose distributed energy efficient role assignment and [38] discusses signal processing techniques to reduce the amount of sensor data needed to detect an event, while [49] proposes the adoption of distributed average consensus in order to compensate sensing errors. In the CASA project [2] researchers adopt techniques for collaborative and adaptive sensing of the atmosphere using radar technologies. All these projects present techniques for static and not mobile sensor networks. To the best of our knowledge, there is little or no work addressing the issue of how to leverage context-sensitive mobile sensing devices such as mobile phones as proposed by Darwin. There is work on the context of using embedded sensors such as the Intel MSP [14] to infer people's activity. However, no interactions between these devices are taken into account in order to realize co-operative strategies such as those discussed in this paper.

Recently, techniques that leverage heterogeneous sensing devices in order to exploit external sensing modalities as a further input for classification algorithms or boosting application fidelity in mobile sensing scenarios are proposed in [25, 15]. Our work goes beyond the idea of borrowing sensor readings from other sensors since we propose collaborative inference techniques that combine with classifier evolution and model pooling.

Semi-supervised machine learning techniques are investigated for word sense disambiguation [50], to identify subjective nouns [41], or to classify emotional and non emotional dialogues [28]. However, no work studies semi-supervised learning techniques in the context of mobile sensing applications or frameworks.

Audio analysis for speaker identification is a well explored area in the literature [17, 42, 16, 44, 43, 26, 18, 40, 23]. Although we do not propose new speaker recognition techniques, we show how to build a lightweight speaker identification application capable of running on mobile phones.

7. CONCLUSION

In this paper we presented the design, implementation, and evaluation of the Darwin system that combines classifier evolution, model pooling, and collaborative inference for mobile sensing applications on phones. The classifier evolution method presented in this paper is an automated approach to updating models over time such that the classifiers are robust to the variability in sensing conditions and settings common to mobile phones. Mobile phones exchange classification models whenever the model is available from another phone, thus, allowing mobile phones to quickly expand their classification capabilities. Collaborative inference combines the classification results from multiple phones to achieve better inference accuracy and confidence. We implemented Darwin on the Nokia N97 and Apple iPhone in support of a proof-of-concept speaker recognition application. We also showed the integration of Darwin with some demo applications. Our results indicate that the performance boost offered by Darwin is capable of off-setting problems with sensing context and conditions and presents a framework for scaling classification on mobile devices. Future work will consider duty cycling techniques for better energy conservation and studying simplified classification techniques, for example, building more computationally light GMMs for mobile phones without impacting performance. We believe the development of such a classification toolkit for mobile

phones will enable new research on phones for human centered applications.

8. ACKNOWLEDGMENTS

This work is supported in part by Nokia, Intel Corp., Microsoft Research, NSF NCS-0631289, and the Institute for Security Technology Studies (ISTS) at Dartmouth College. ISTS support is provided by the U.S. Department of Homeland Security under award 2006-CS-001-000001, and by award 60NANB6D6130 from the U.S. Department of Commerce. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of any funding body. A special thanks goes to our shepherd Deborah Estrin.

9. REFERENCES

[1] Apple iPhone. http://www.apple.com/iphone.
[2] The CASA project. http://www.casa.umass.edu/.
[3] FFTW. http://www.fftw.org.
[4] Kiss FFT. http://sourceforge.net/projects/kissfft.
[5] Nokia Energy Profiler. http://tinyurl.com/ycostaq.
[6] Nokia Series. http://europe.nokia.com/nseries/.
[7] Qt platform. http://qt.nokia.com/.
[8] Sensorplanet. http://research.nokia.com/research/projects/sensorplanet/index.html.
[9] T. Abdelzaher, et al. Mobiscopes for Human Spaces. In *IEEE Pervasive Computing*, volume 6, page 20. IEEE, 2007.
[10] M. Azizyan, et al. SurroundSense: Mobile Phone Localization via Ambience Fingerprinting. In *MobiCom'09*, 2009.
[11] S. Basu. A linked-HMM Model for Robust Voicing and Speech Detection. In *ICASSP'03*, 2003.
[12] A. Campbell, S. Eisenman, N. Lane, E. Miluzzo, and R. Peterson. People-Centric Urban Sensing. In *WICON'06*, 2006.
[13] A. Campbell, et al. The Rise of People-Centric Sensing. In *IEEE Internet Computing Special Issue on Mesh Networks*, July-August 2008.
[14] T. Choudhury, et al. The Mobile Sensing Platform: an Embedded System for Capturing and Recognizing Human Activities. *IEEE Pervasive Magazine, Spec. Issue on Activity-Based Computing*, 2008.
[15] S. B. Eisenman. People-Centric Mobile Sensing Networks. *Ph.D. Dissertation*, October 2008.
[16] H. Ezzaidi and J. Rouat. Pitch and MFCC Dependent GMM Models for Speaker Identification systems. In *Electrical and Computer Engineering, 2004. Canadian Conference on*, volume 1, 2004.
[17] Z. Fang, Z. Guoliang, and S. Zhanjiang. Comparison of Different Implementations of MFCC. *J. Comput. Sci. Technol.*, 16(6), 2001.
[18] G. Friedland and O. Vinyals. Live Speaker Identification in Conversations. In *ACM Multimedia'08*, 2008.
[19] R. Honicky, et al. Increasing the Precision of Mobile Sensing Systems Through Super-Sampling. *UrbanSense08*, page 31, 2008.
[20] R. Honicky, et al. N-SMARTS: Networked Suite of Mobile Atmospheric Real-Time Sensors. In *Networked Systems for Developing Regions'08*, 2008.
[21] A. Kansal, M. Goraczko, and F. Zhao. Building a Sensor Network of Mobile Phones. In *IPSN'07*.
[22] D. Kim, J. Hightower, R. Govindan, and D. Estrin. Discovering Semantically Meaningful Places from Pervasive RF-Beacons. In *UbiComp '09*, pages 21–30. ACM, 2009.
[23] T. Kinnunen, T. Kilpeläinen, and P. Fränti. Comparison of Clustering Algorithms in Speaker Identification. In *SPC 2000*.
[24] R. Kumar, et al. DFuse: a Framework for Distributed Data Fusion. In *SenSys'03*, 2003.
[25] N. Lane, et al. Cooperative Techniques Supporting Sensor-Based People-Centric Inferencing. In *Pervasive'08*.
[26] I. Lapidot, H. Guterman, and A. Cohen. Unsupervised Speaker Recognition Based on Competition Between Self-Organizing Maps. *IEEE Transactions on Neural Networks*, 13(4):877–887, 2002.
[27] H. Lu, et al. SoundSense: Scalable Sound Sensing for People-Centric Applications on Mobile Phones. *MobiSys'09*, 2009.
[28] B. Maeireizo, D. Litman, and R. Hwa. Co-Training for Predicting Emotions with Spoken Dialogue Data. In *ACL'04*, 2004.
[29] E. Miluzzo, et al. Sensing Meets Mobile Social Networks: the Design, Implementation and Evaluation of the CenceMe Application. In *SenSys'08*, 2008.
[30] P. Mohan, et al. Nericell: Rich Monitoring of Road and Traffic Conditions Using Mobile Smartphones. In *SenSys'08*, 2008.
[31] M. Mun, et al. PEIR, the Personal Environmental Impact Report, as a Platform for Participatory Sensing Systems Research. In *MobiSys'09*, 2009.
[32] M. Musolesi, et al. Supporting Energy-Efficient Uploading Strategies for Continuous Sensing Applications on Mobile Phones. In *Pervasive'10*, 2010.
[33] S. Nath, et al. Synopsis Diffusion for Robust Aggregation in Sensor Networks. In *SenSys'04*, 2004.
[34] Nokia. N900. http://maemo.nokia.com/n900.
[35] NRC. Sensing the World with Mobile Devices. *Technical Report*, 2008.
[36] R. Olfati-Saber. Distributed Kalman Filter with Embedded Consensus Filters. In *CDC-ECC'05*, pages 8179–8184, 2005.
[37] R. Olfati-Saber. Distributed Kalman Filtering and Sensor Fusion in Sensor Networks. *Lecture notes in control and information sciences*, 331:157, 2006.
[38] S. Patil, et al. Serial Data Fusion Using Space-Filling Curves in Wireless Sensor Networks. In *IEEE SECON'04*, 2004.
[39] D. Reynolds, et al. Speaker Verification Using Adapted Gaussian Mixture Models. *Digital Signal Processing*, 10(1-3):19–41, 2000.
[40] D. Reynolds and R. Rose. Robust Text-Independent Speaker Identification Using Gaussian Mixture Speaker Models. *IEEE transactions on Speech and Audio Processing*, 1995.
[41] E. Riloff, J. Wiebe, and T. Wilson. Learning Subjective Nouns Using Extraction Pattern Bootstrapping. In *CoNLL'03*, 2003.
[42] E. Shriberg. Higher-Level Features in Speaker Recognition. *Lecture Notes in Computer Science*, 4343:241, 2007.
[43] E. Shriberg, et al. Modeling Prosodic Feature Sequences for Speaker Recognition. *Speech Communication*.
[44] E. Shriberg and E. Stolcke. The Case for Automatic Higher-Level Features in Forensic Speaker Recognition. In *Interspeech'08*, 2008.
[45] N. Shrivastava, et al. Medians and Beyond: new Aggregation Techniques for Sensor Networks. In *SenSys'04*, pages 239–249. ACM New York, NY, USA, 2004.
[46] D. Siewiorek, et al. Sensay: a Context-Aware Mobile Phone. In *ISWC'03*, 2003.
[47] C. Tadj, et al. GMM Based Speaker Identification Using Training Time Dependent Number of Mixtures. In *ICASSP'98*, volume 2, 1998.
[48] Y. Wang, et al. A Framework of Energy Efficient Mobile Sensing for Automatic User State Recognition. In *MobiSys'09*, 2009.
[49] L. Xiao, S. Boyd, and S. Lall. A Scheme for Robust Distributed Sensor Fusion Based on Average Consensus. In *IPSN'05*, 2005.
[50] D. Yarowsky. Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. In *ACL'95*.
[51] S. Zhang, S. Zhang, and B. Xu. A Robust Unsupervised Speaker Clustering of Speech Utterances. In *IEEE NLP-KE'05*.
[52] J. Zhao, et al. Computing Aggregates for Monitoring Wireless Sensor Networks. In *SNPA'03*, 2003.
[53] M. Zhu. Recall, Precision and Average Precision. In *Technical Report, University of Waterloo*, 2004.
[54] X. Zhu. Semi-Supervised Learning Literature Survey. In *Computer Sciences TR, University of Wisconsin-Madison*, 2008.