A Comprehensive Approach to Participatory Sensing of Weather Information via Mobile Devices

Amr Elsaadany
Computer Engineering Dept.
Pharos University in Alexandria

ABSTRACT
Data sensing techniques are becoming widely used in various applications including forecasting systems. Accurate forecasting systems must rely on multiple input data sources. In this paper, the techniques used in developing accurate weather reporting systems are reviewed and the strength of multiple data sensing techniques is utilized to conceptualize a new system architecture that aims at accurate weather forecasting. The new model is based on four main components; environmental sensing component, user submitted reports, social networks forecast, and external sensors components. The resulting system produces more accurate reports than systems that do not rely on multiple input sources.

General Terms
Pervasive Computing, Mobile Computing, and Applications of Computer Software in Forecasting.

Keywords
Sensors networks; crowd sensing; mobile sensing; participatory sensing; weather forecast

1. INTRODUCTION
Accurate weather observations are important for meteorology studies, energy planning, transportation, and manufacturing. However, the data accuracy is quite limited by the low concentration and wide dispersion of weather sensing stations around the world. There are many weather checking software tools available, however, their methods rely on one of the techniques for weather forecasting [1]. Consequently, there is a need to enhance the forecast accuracy based on multiple data sources, many of which depend on the wide spread of mobile computing and the ability to sense the environmental variables under consideration.

Currently, many smartphones are equipped with environmental sensors like temperature, humidity, and pressure. The utilization of these sensors for improving weather forecasting models is becoming a research interest given the widespread these smartphones equipped with such sensors. As such, this paper proposes a multi-faceted system that utilizes the sensors already present in smartphones; collects human-input observations of the weather around them; analyzes public weather reports posted to social networks; and uses external sensors connected to smartphones for additional sensory input. By aggregating these methods together, and identifying which methods are best for which use-cases, a more accurate weather forecasting model can be created.

The first section of the paper overviews the technical solutions for data gathering and forecasting. While most of these solutions adopt one method or another for data collection, the proposed approach depends on multiple input points among which is the sensing of various weather observations, like temperature, humidity and pressure. The proposed system consists of four sub-systems or components that collectively help create a more accurate real-time weather monitoring system compared to what is available today. The system components are described in the requirements sub-section below. The paper is organized as follows: the related work and technologies are summarized in Section 2. Section 3 describes the system architecture and Section 4 describes the system implementation strategy. The discussion and the conclusion are given in the last section.

New System Requirements

The first proposed component of the system will rely on automatically reading the sensor data from the user smartphones in the background without user interaction, and sending this collected data to a server that aggregates, filters and stores this information. To maintain a high accuracy of the automatically collected information, like temperature reading from a smartphone ambient temperature sensor, the system must collect several other pieces of information, for example, to identify whether that reading was captured indoor or outdoor. Presence of a known Wi-Fi network, and a weak GPS signal, can imply an indoor environment, which calls for not using the captured data. Moreover, the movement combined with low proximity and low ambient light as capture by a smartphone's ambient light sensor, can imply the smartphone being in a user's pocket, hence another reason to disregard the captured information.

The second component of the system will rely on user input and validation, to collect weather condition information that are not easily possible to detect in software or hardware, like the cloud coverage, and whether or not it's raining or snowing. To maintain a high accuracy of the user-submitted information, a rating system is employed that ranks user submissions (e.g., cloud coverage). The system uses historical data about the accuracy of this user's past submissions, plus submissions from other users in close vicinity, to estimate the likelihood of accuracy of one submitted piece of information.

The third component of the system automatically searches and collects weather information reported by all users on social networks. Many users of social networks, like Twitter, post status updates that reflect how the weather is around them. They post messages like "it's raining now in NY" or "I love this 80 degree afternoon." Since many of the messages are also geo-tagged, natural language processing can be used to extract the weather-related information from these messages. The challenge in the third component of the system is parsing the collected messages for meaningful and valuable weather-related information, taking care of possible spelling mistakes, observation inaccuracies, uncertainty about location exactness, and filtering out generic statements like "I hate when it's 100 degrees hot."

The fourth and final component of the system compromises a hardware module that connects wirelessly via Bluetooth to a smartphone. This module is equipped with environmental sensors like temperature, humidity, pressure, air pollution, radiations and Carbon Monoxide, and can stream the readings in real-time to the smartphone. The advantage of an external module is to provide a wider range of environmental observations than is possible with most smartphones. While it is true that an external module is more difficult to carry than a smartphone; the plan is to create an extremely compact solution that can be carried by the user (as part of a keychain).

The system also has a smartphone application, which supports all four sub-systems of smartphone sensor reading, human input collection, social network collection, and external sensor reading. This application aggregates all or some of the readings of these sub-systems, and sends them to a remote server. The server then cross-examines the readings, and combines them with publicly available weather information from public weather stations to produce new weather forecasting models. The details of the application modules will be presented in later sections.

2. BACKGROUND AND RELATED WORK

Several research works have targeted the advantages of various forms of technologies in the data sensing and forecasting field. In this section, the existing techniques and platforms that can support wide range of sensing applications are described. This background review helps set the stage for the usage of new approaches such crowd sensing data collection and analyses via mobile applications.

ResearchKit [2] is an open source software framework to allow developers to create research applications. It leverages the sensors of the iPhone to track movement, take measurements and record data. The kit consists of three modules, the first of which is a Survey module that provides pre-built user interface elements to easily collect answers to researcher-specified questions. The second module is the Informed Consent, which is used to clarify what needs to be provided and who will have access to the information. The third module is the Active Tasks, which allows inviting users to perform activities under semi-controlled conditions, while iPhone sensors actively collect data. The Active Tasks includes Gait (which uses the Accelerometer and Gyroscope), Tapping (which uses the touch screen), and others that collects motion, location, heart rate and other activities.

Atmos is a mobile-based platform that collects sensory data through the mobile phones' available on-board sensors, along with human input. The authors employed the available sensor found in a mobile device such as environmental pressure, temperature, luminosity, and humidity. Additionally, they query the application users to enter current and future weather conditions. Collected data is then uploaded, processed and clustered by location in an online database. Users can access these data by searching for a weather report for a particular location. It was possible to collect 18,000 sensor measurements and 500 human inputs. The authors indicated a significant positive relationship between recorded battery temperature and pressure [3].

Autorasaurus was designed for the purpose of increasing the accuracy and timeliness of Aurora Forecast, which provides real time weather services. The data collection is based on the use of the vast data already available on Twitter as input. The system uses the Twitter API to go through all geo-coded tweets mentioning Aurora sightings and other interested users can verify these sightings. The users of the application can manually input a sighting directly [4].

A system was designed in [5] for finding weather information reported on Twitter. An experiment was conducted at the city of Pretoria, South Africa. Every day before midnight, two Twitter search queries are performed. The first looks for tweets tagged with relevant terms as #ptaweather and #pretoria, and the second query searches for tweets geo-tagged in Pretoria that have terms like storm, rain, cold, hot, sun, etc. The tweets are separate into two groups; the first is called "ground truth" which are tweets from organizations that possess detailed structure weather reports. The second group is called "public category" which includes weather tweets from the public and can be casual in nature like "Gloomy weather today" or "Good Morning it's such a beautiful day." The system utilizes a model that spots the predefined topics in these tweets then process them and analyze the results. The algorithm successfully classified 85% of the tweets.

To mitigate the lower accuracy of temperature observations inside cities, the authors in [6] developed an Android application that reads the phone's battery temperature and uses a heat transfer model to estimate daily mean air temperatures. The authors collected 220 million battery temperature readings in one year, and carried out their analysis on a 2.1 million reading subset from 8 major cities. They then averaged these temperatures in space and time to obtain daily averages for each city. The officially recorded daily mean temperatures as measured by airport-based weather stations were collected and utilized to calibrate a heat transfer model and for validation purposes. The authors found that the mean absolute error of such measured daily air temperatures amounts to 1.45 degree Celsius. The authors believe that, by averaging over a sufficiently large number of battery temperature readings, the existing variation in thermal conductivity over individual readings is adequately filtered. Even though some readings were captured indoors, the influence of outside air temperature was already sufficiently reflected in the battery temperature readings.

In [7], the authors present the design and implementation of a portable measurement device for measuring air pollution by connecting a low-cost ozone sensor to a smartphone running the Android OS. The hardware sensor communicates with the RS232-TTL interface to the smartphone via USB. It is powered by an external array of four AAA batteries, which is estimated to have a lifetime of 50 hours of active sensing given a highest measured current draw of 50mA. The authors observed that low-cost gas sensors, like the one used, must be frequently re-calibrated, so they implement a re-calibration system that takes into account available reference measurements from static reference stations maintained by official authorities. During the measurements, the authors mounted the sensor on a bicycle and took measurements from several rides around the city. Knowing that the daily ozone concentration typically ranges between 0 and 70 ppb, the observed mean error from the system was 2.74 ppb.

HazeWatch [8] is a low-cost air pollution-sensing device users can mount to their vehicles, coupled with a mobile app that tags the sensed data with location and time and uploaded it to a server, a server that stores the data and applies interpolation models to generate spatio-temporal estimates, and visualization tools that map pollution levels. In their deployment of the system for over 2 years, the authors found that metal oxide sensors are cheap, but are non-linear and unreliable, while the electrochemical sensors are expensive

and extremely sensitive. They also found that the calibration of the sensor units is challenging, and that the packaging and mounting presented some difficulties.

In [9], the data collected by smartphones sensors used to try to get accurate weather forecasting. However, the accuracy of the data collected with the mobile sensors affected by external parameters and it is not possible to calculate the correction constant that increases the accuracy of the sensed data.

CrowdSearch [10] presents a mobile-based search solution that combines automated image-matching with real-time human validation of search results. The solution claims a 95% precision response-time within minutes and with low cost per search query. In this system, delay-accuracy-cost models for crowd-sourced users are developed to decide which images need validation and estimates the delay of responses from human validators, and how to price the validation tasks. The system saves monetary cost by up to 50% in comparison with non-adaptive schemes. The system incurs only 5-6% larger delay in comparison to a delay-optimal scheme. The dynamic partitioning of search across mobile phones and remote servers saves overall energy consumption up to 70%. Instead of using a parallel crowd sourced posting for result validation (which sacrifices cost), or a serial posting (which sacrifices delay), the proposed algorithm balances these two together. Given a deadline for results, the algorithm waits for the first responses to validate the first candidate search result, and if by a certain pre-determined time, it can calculate that the probability of the upcoming validation tasks are irrelevant given the current received results and past historical data, it halts further validation and returns the result.

In [11], the authors look into the obstacles facing large-scale adoption of crowd sensing. They identify these to be; 1) the heterogeneity of sensing hardware and mobile platforms, 2) the users have to install a separate proprietary application for every crowd sensed experiment in which he/she wishes to participate, and 3) the increasing network bandwidth demands. The authors present a solution based on 1) separation of data collection and sharing from application-specific logic, 2) removal of application installation on smartphones from the critical path of application deployment, and 3) decentralization of processing, and data aggregation near the source of data. In their proposed solution, the mobile devices are reduced to a role of forwarding sensor data to proxy Virtual Machines, which comprises a distributed cloud infrastructure deployed close to the users. Each proxy VM is associated with a single smartphone and is kept physically close to the user through VM migration. This proxy VM in turn forwards the requests from mobile devices to application VMs that perform the data processing. These application VMs are managed and deployed by a single application server running on the centralized cloud infrastructure.

A research into how to better select participants in a mobile crowd sensing application to minimize the user's incentive payments while satisfying probabilistic coverage constraints is covered in [12]. The paper states that the sensing coverage in mobile crowd sensing applications relies on the uncontrollable mobility of people, and thus it is important to consider their mobility patterns. However, finding full coverage is not always required, and it is sufficient to ensure a high ratio of spatial coverage in a specified period. Additionally, uploading sensed results in parallel with a 3G call can reduce about 75% of energy consumption. The work focuses on selecting the minimal number of participants in crowd sensing under probabilistic coverage constraints, with consideration of both total energy consumption and incentives paid per task. The application achieves fewer participants on average than the baseline, under the same coverage constraints.

Medusa [13] is a high-level programming framework for easily writing task descriptions for crowd sensing tasks. It employs a distributed runtime system that coordinates the execution of these tasks between smartphones and a cluster in the cloud. The authors set a number of requirements that their system should have: 1) Requestors must be able to specify worker-mediation, like a worker needing to perform an action to complete a stage such as initiating the recording of a video clip. 2) Requestors must be able to specify monetary incentives for workers. Some tasks may also require reverse-incentives, where the workers pay the requestors for the privilege of participating in the task. 3) Tasks may have timeliness requirements and any contribution received after the deadline is discarded. 4) Workers must be able to sign up for multiple concurrent tasks, and requestors should be able to initiate multiple tasks concurrently. The runtime should preserve subject anonymity with respect to requestors, and should contain mechanisms for ensuring data privacy. The Medusa framework uses MedScript which is an XML-based language that consists of two high-level abstractions: stages, which describe a sensing or computation action, and connectors, which express control flow between stages. The script is interpreted in the cloud by the MedScript interpreter. Then, a Task Tracker spawns as many instances as needed for the require worker stages. It coordinates the execution of every stage and maintains instance state information in persistent storage. A single worker may concurrently sign up for multiple instances of the same task, and/or instances of many different tasks. The Task Tracker does not know the identities of the requestor or the worker, instead referring to them in its internal data structures using a non-transparent machine-generated ID. In evaluating Medusa, the authors implemented a number of applications such as data documentation, forensic analysis, and road monitoring.

Nericell [14] is a system for efficiently monitoring road and traffic conditions, which is specially tailored for developed countries. The authors observe that road and traffic conditions in developed countries are much more complex and uncertain than other countries for which most of the research had been performed previously. The system uses various sensors such as accelerometers and microphones. The system uses an algorithm for determining the accelerometer orientation and then re-orient automatically as needed. However, any extraneous acceleration that occurs while the user is interacting with the phone must be neglected.

Another participatory sensing through passengers' smartphones is utilized in [15] to predict bus arrival times. The system extracts unique identifiable fingerprints of public transit buses and utilizes the microphones on mobile phones to detect the audio indication signals of bus card readers. It also leverages the accelerometer of the phones to distinguish the travel pattern of buses to other transport means. Then, based on both historical knowledge and the real-time traffic conditions the system predicts the bus arrival time of various routes. The system includes a database that stores sequences of cell-tower IDs that are experienced along different bus routes. To detect whether the user is in a public transit bus or using a non-public bus, the authors employed a low-energy solution to auto-detect the short beep audio responses from the card readers since these audio responses are distinct to public transportation methods. The authors observe that such system's accuracy is limited by the number of participating passengers, and that the system is less accurate in the first few

bus stops due to the lack of a sufficiently long cell-tower sequence to accurately classify the route.

In [16], the authors propose a system for participatory urban sensing of environmental noise pollution in urban areas, as an alternative for the existing models that are passed on population and traffic models rather than on real data. The system crowd sources the collection of environmental data in urban spaces to people, who carry smart phones equipped with sensors and GSP receivers. The paper proposes a solution to the problem of incomplete samples by using data sensing that focus on roadside noise pollution. The authors showed that they could recover a noise map with high accuracy, allowing nearly 40% missing samples while reducing communication costs by 30%.

In view of the above mentioned background, it is becoming clear that there is a good amount of work being done in the area of participatory sensing and how they can be used to try getting better sensing data for a variety of applications. It is quite possible to apply some of these techniques to weather data collection and forecasting with the aim of improving the accuracy of weather reporting. The question is how to employ some of these techniques to build a platform for a more accurate forecast and this question is addressed in the rest of the paper.

3. SYSTEM ARCHITECTURE

This section presents the architecture of the proposed system and illustrates how the various components interact with each other. As seen in Figure 1, the architecture consists primarily of a client-side mobile application running on compatible smartphones, and a forecast server that runs the necessary services for forecast data collection, filtering, and distribution.

The mobile application consists of four modules. The first module is the Environmental Sensing Module which identifies the hardware capabilities of the smartphone and its available environmental sensors, like the presence of a barometer sensor for example, and uses the smartphone operating system's Application Programming Interfaces (APIs) to collect readings from these sensors at specific intervals.

The second module is the User-Submitted Reports component. This component provides the end-user the capability to submit their own observations of the weather around them, or the forecast as reported by their local weather network. Some of the details that could be reported by the user are hard to measure, like the "current cloud coverage" or the "raining situation" and these reports have to be filtered by the system.

The third module is the Social Networks Forecast Collector. This module searches Social Networks such as Twitter and Facebook for weather-related status messages, analyzes these messages, and extracts the meaningful weather information.

The last module of the mobile application is the External Sensors component. This component connects wirelessly to external devices that have additional environmental sensors that may not be present in a smartphone like Air Quality sensors, and fetches environmental readings from them.

The mobile application may collect forecast data from one or more of these modules. As introduced before, multiple sources improve the accuracy of the forecast, and this is the target of the proposed system architecture. The usage of multiple modules depends on the capabilities of the smartphone, the user's willingness to submit reports, whether or not the user links his social accounts, and whether or not the user is in possession of the compatible sensors.

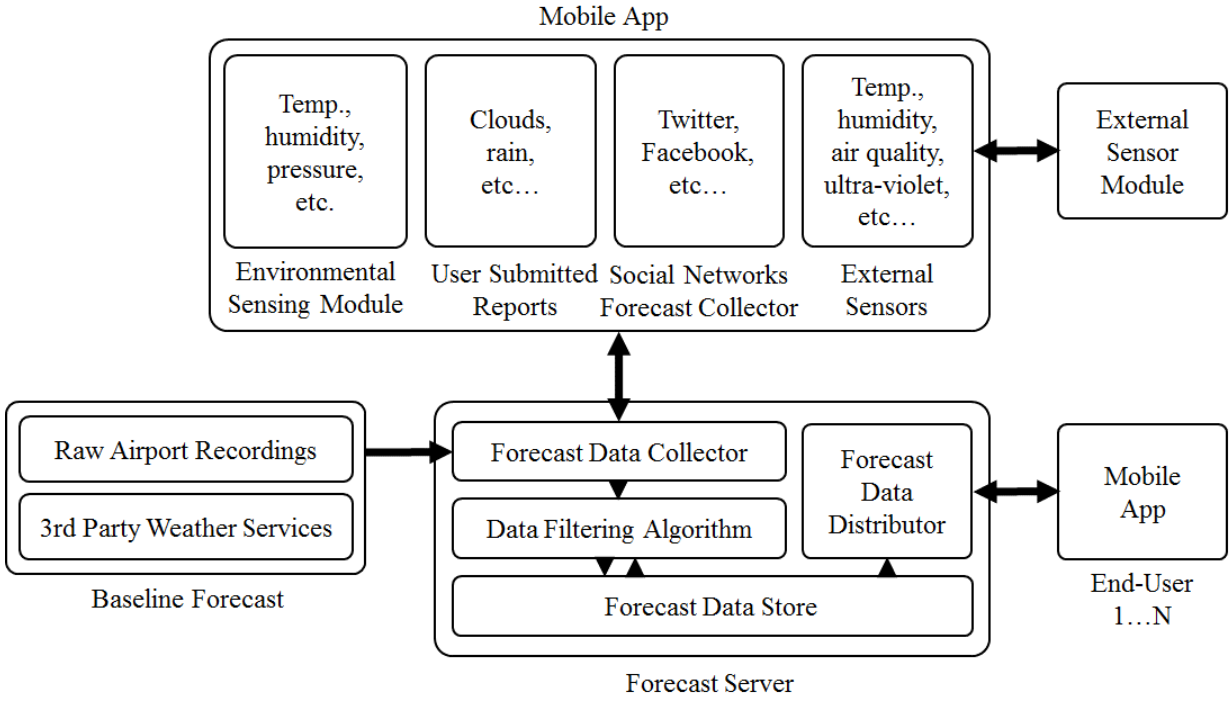

Fig 1: System Architecture

4. SYSTEM IMPLEMENTATION STRATEGY

All of the collected environmental weather data is sent at a preset interval to the Forecast Server, where they are received by a Forecast Data Collector. They are then sent through a Data Filtering Algorithm, which takes into account the location, the forecast reports of other users, the past accuracy of the user's submitted reports, and the Baseline Forecast data to adjust these readings. The adjusted data, along with the original data, are then stored in the Forecast Data Store.

The Forecast Data Distributor is the Mobile Application's gateway to the Forecast Data Store. It sends back to the End User the forecast data for his requested location, as filtered and combined from the several data sources available.

The Baseline Forecast component connects to third party weather services to retrieve the weather forecast. It also collects raw weather data readings from weather sensing stations (usually at airports). These two types of forecast data are then used to augment the collected weather data and compare its accuracy.

For the Environmental Sensing, the available sensory devices, available in the smartphone can be queried. The query depends on the smartphone and if it has a built-in sensors or an external ambient temperature sensor is attached to it. For each of the available environmental sensors, the current readings are stored and then sent to the Forecast Server.

The Forecast Server is responsible for the collection, filtering, storage and distribution of weather information. It collects readings from the mobile application installed on user's smartphones, collects weather forecasts from third party web services, and distributes forecasts to end-users through a web services that report the weather for the requested locations.

4.1 Details of System Implementation

Basically, the implemented system consists of a mobile application and a web based back end system. The back end is done using Java, the Spring web model-view-controller framework, the Hibernate object-relational mapping framework, and MySQL database. The application senses the temperature periodically and send it through an HTTP request to the back end system. Figure 2 shows the implementation of the system.

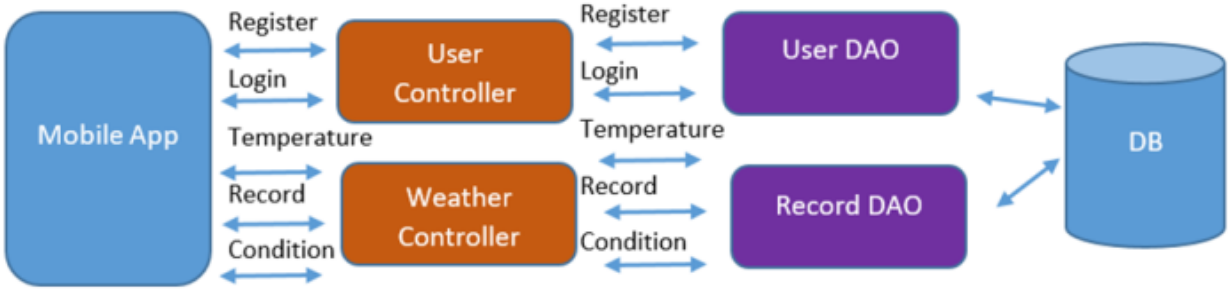

Fig 2: System Implementation Diagram

The system provides the average temperature of the sensed data of all users within the same city and within one hour. A sample data is shown in Table 1. The system data shows that the average temperature has 2.5% accuracy over the web forecast given over the internet (via Google in this case).

Timestamp	Sensed Temperature	External Temperature
5:31:30	22	23
6:29:51	20	22
7:30:50	19	21
8:30:40	20	20
9:30:43	19	19
10:33:06	20	18
Average	20.00	20.50

Table 1: A sample data comparison

The back end system contains Plain Old Java Objects (POJOs) that maps the database tables, in addition to the Data Access Objects (DAOs). The DAOs used by Hibernate framework to access the database using HQL query language. This approach makes the parts of the system loosely coupled and grantees resiliency and maintainability of the system. The DAOs are accessed by Spring controllers that accept HTTP request from any web client interface with JSON payload.

5. CONCLUSION

After reviewing the available data collection and forecasting techniques, a new architecture is proposed based on multiple input data points in order to establish a comprehensive local weather forecasting system. The new approach relies on four input components to enable the system to get a more accurate weather forecasting that is not possible with other systems. The first component, Environmental Sensing, queries the available sensory devices on the smartphone. The second component, User Submitted Reports, provides the end-user the capability to submit their own observations of the weather around them, or the forecast as reported by their local weather network. The third component, Social Networks Forecast, searches Social Networks for weather-related status messages, analyzes these messages, and extracts the meaningful weather information from them. The last component is the External Sensors component, which connects to external devices that have additional environmental sensors that are not present in a smartphone and fetches environmental readings from them.

The implemented system prototype, used the sensing capabilities of the mobile devices and collected periodic weather information along with textual description of the weather (e.g. cloudy, rainy, etc.). The results show a more accurate weather data than those available to users from other web sources. Further enhancements to increase the geographical area of the forecast is now being considered for obtaining even more conclusive report about the weather condition. As each input component has some weaknesses and strengths, combining the input data from all the system components enables the use of each sub-system at its strengths.

6. ACKNOWLEDGMENTS

This is to acknowledge Eng. Amr Ramadan for his contribution to the early part of this work.

7. REFERENCES

[1] Wankhede, P., Sharma, R., Pote, C. 2014. A Review on Weather Forecasting Systems Using Different Techniques and Web Alerts. International Journal of Advanced Research in Computer Science and Software Engineering, Volume 4, Issue 2.

[2] developer.apple.com, Apple Researchkit. [Online]. Available: http://developer.apple.com/ [Accessed: 15-Jul- 2016].

[3] Niforatos, E. et al. 2014. Atmos: A Hybrid Crowdsourcing Approach to Weather Estimation. Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing Adjunct Publication - UbiComp.

[4] LaLone, N. et al. 2015. Harnessing Twitter and Crowdsourcing to Augment Aurora Forecasting. Proceedings of the 18th ACM Conference Companion on Computer Supported Cooperative Work & Social Computing - CSCW'15 Companion.

[5] Butgereit, L. 2014. Crowdsourced weather reports: An implementation of the μ model for spotting weather information in Twitter. IST-Africa Conference Proceedings.

[6] Overeem, A. et al. 2013. Crowdsourcing Urban Air Temperatures from Smartphone Battery Temperatures. Geophysical Research Letters 40.15: 4081-4085.

[7] Hasenfratz, D., et al. 2012. Participatory Air Pollution Monitoring Using Smartphones. 2nd International Workshop on Mobile Sensing.

[8] Sivaraman, V., et al. 2013. Hazewatch: A Participatory Sensor System for Monitoring Air Pollution in Sydney. 38th Annual IEEE Conference on Local Computer Networks – Workshops.

[9] Dalğın, S. and A. Doğru, O. 2015. Investigation of the Usability of Mobile Sensors for Weather Forecasting. International Journal of Environment and Geoinformatics.

[10] Yan, T., Kumar, V. and Ganesan, D. 2010. CrowdSearch- Exploiting Crowds for Accurate Real-time Image Search on Mobile Phones. Proceedings of the 8th international conference on Mobile systems, applications, and services - MobiSys.

[11] Xiao, Y. et al. 2013. Lowering the Barriers to Large-Scale Mobile Crowdsensing. Proceedings of the 14th Workshop on Mobile Computing Systems and Applications - HotMobile.

[12] Zhang, D. et al. 2014. Crowdrecruiter: Selecting Participants for Piggyback Crowdsensing under Probabilistic Coverage Constraint. Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing - UbiComp.

[13] Ra, M. et al. 2012. Medusa: A Programming Framework for Crowd-Sensing Applications. Proceedings of the 10th international conference on Mobile systems, applications, and services - MobiSys.

[14] Mohan, P., Padmanabhan, V., and Ramjee, R. 2008. Nericell: Rich Monitoring of Road and Traffic Conditions using Mobile Smartphones. Proceedings of the 6th ACM conference on Embedded network sensor systems - SenSys.

[15] Zhou, P., Zheng, Y., and Li, M. 2012. How Long To Wait? Predicting Bus Arrival Time with Mobile Phone based Participatory Sensing. Proceedings of the 10th international conference on Mobile systems, applications, and services - MobiSys.

[16] Rana, R. et al. 2010. Ear-Phone: An End-to-End Participatory Urban Noise Mapping System. Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks - IPSN.