Quality controlling web-based and real-time crowdsourced data

Aaron Steele
Jack Baskin School of Engineering
University of California Santa Cruz
Santa Cruz, CA
atsteele@ucsc.edu

ABSTRACT

In this paper we propose a framework for breaking down and characterizing different parts of quality control in relation to crowdsourced data. Controlling the quality of submissions for crowdsourced tasks on platforms like MTurk is a known problem, and one we believe deserves more research into.

We also propose a new technique for quality control of ubiquitously crowdsourced data. The rise of smartphones has brought about much easier data gathering from the crowd as they go about their daily lives. Applications that gather and process this data, typically for use in real-time applications, are called ubiquitously crowdsourced. The problem with gathering data from such a large group of people—who are typically not evaluated based on their credentials at all—is the ease of which it can be for malicious users to submit bad data to the applications. We propose a new technique that uses the mobility patterns of users, on top of the quality of their prior submissions, to help solving this problem.

1. INTRODUCTION

Crowdsourcing has become a much more popular way of both getting and processing data in recent years. In processing data, crowdsourcing is primarily helpful for tasks which are easy for humans to do but hard to machines. [1] Getting data, on the other hand, is a very different process. With the advent of the ubiquity of smartphones, it has become much easier to gather various types of data from the crowd, as it were.

Quality controlling web-based crowdsourcing is, in a lot of ways, easier than doing the same with ubiquitously crowdsourced data. We have created a taxonomy of quality in web-based crowdsourced systems. There are various approaches to quality controlling the processed data, which we will discuss.

Following the overview of current ideas for quality controlling web-based crowdsourcing, we will explain the idea and formula we have created for controlling quality of ubiquitous crowdsourcing. Ubiquitous crowdsourcing presents different problems than web-based does, such as dealing with real-time events, and the dynamic crowd.

2. BODY

2.1 Web-based Crowdsourcing

Crowdsourcing has been on the rise in recent years. Websites like Wikipedia and StackOverflow are prominent in their use of crowdsourcing. These websites provide excellent services, and show how powerful properly leveraged and utilized crowdsourcing can be. Of course, an inherent factor in these websites is that the contributors might be of wildly differing levels of knowledge or skill on any given topic. This means the quality of any given submission cannot be assumed to be correct or up to the standard the developers require. Which means all user submissions must be controlled for quality. Quality controlling is also required for another reason: malicious submissions. It is known that the guise of anonymity can bring out the worst in many people (just read through the comments of a website like Youtube.) This means that often people will attempt to submit malicious or incorrect data on purpose, to bring down the overall quality of the service, or even try to totally discredit the service as valuable.

To talk about quality control we must define and categorize it. First, the creator of the task, or the *requestor*, goes on to a given crowdsourcing platform and creates a task. That task could be asking a question on StackOverflow, or using MTurk to have people fill out a survey. The *workers* do the task and submit their contributions via the crowdsourcing platform. Then, once all or some of the results are in, the requester assesses the quality of the submissions. Quality is, of course, subjective to each requester.

For this paper we will be using Crosby's definition of quality. [2] He emphasizes 'conformance to requirements' as the best principle to design quality control models with. This

Conference: UbiComp'11 (https://dl.acm.org/citation.cfm?id=2030100&picked=prox)

1: A. J. Mashhadi and L. Capra, "Quality control for real-time ubiquitous crowdsourcing," in Proceedings of the 2nd international workshop on Ubiquitous crowdsouring, pp. 5-8, ACM, 2011.

2: M. Allahbakhsh, B. Benatallah, A. Ignjatovic,H. R. Motahari-Nezhad, E. Bertino, and S. Dustdar, "Quality control in crowdsourcing systems:Issues and directions," IEEE Internet Computing,vol. 17, no. 2, pp. 76-81, 2013.

ACM ISBN 123-4567-24-567/08/06.
DOI: 10.475/123_4

means we define the standard of quality for submissions by "the extent to which the provided outcome fulfills the requirements of the requester."

We then characterize the quality of submissions based on two main factors: worker profiles and task design. We have proposed a taxonomy for quality in crowdsourcing systems, as fig. 1 shows.

A worker's abilities can effect the submissions to a large degree. [3] As seen in fig. 1, a worker's profile consists of two elements: their reputation and expertise. The higher a worker's expertise is, the higher their submissions quality is expected to be. The elements are separate, reputation is a more general score than expertise. Reputation is also considered public, anyone on the site can view it, whereas expertise might not be. Expertise is also job-dependent; for example a worker who has a high C++ expertise might not be be suitable for a SQL task.

- Reputation is a measure of reliability, or a trust score, between a worker and a requester. The higher the score, the more likely the worker is to submit quality work, which means the requester can expect higher quality work from a worker with a high reputation. Reputation is primarily gained or lost from other members of the community rating submissions. [4] On websites like StackOverflow this feedback is explicit, users can upvote or downvote a answer or comment. On other websites, like Wikipedia, the feedback is implicit. For example how often other, later, editors keep the worker's submission without changing it, would be an excellent implicit method of calculating reputation.
- Expertise is a measure of how capable a worker is of doing a certain task. [5] Expertise can be further broken down into credentials and experience. Credentials are outside accreditations, such as academic degrees, technical certifications, or other easily-verified attributes. Experience is, as it suggests, prior experience on the crowdsourcing platform. Tasks completed, work submitted, all of it contributes to the experience of a given worker.

Task design is how a requester breaks down and specifies the work workers are supposed to complete. We have identified four properties of task design that contribute to the quality of submissions: task definition, user interface, granularity, and compensation policy. (See fig. 1.)

- Task definition is the information the requester gives to the workers to define what, exactly, they want the workers to do. Firstly, the requester supplies a short description of the task to be done, the time limits, etc. [6] Second, the qualifications and expertise are listed. Anything that limits the worker is here. For example, if the requester only wanted people to take the survey who lived in the UK, or had academic degrees in Sociology. Prior studies show that a high quality task definition leads to higher quality submitted work. [6]
- The user interface is how all workers submit their work, or interface with the task. This interface can be a standard web UI, or something more complicated like an API. The balancing act in creating a UI is between too few restraints or conditions and too many. Too few and workers will find an easy way to cheat and enter more general data, making their submissions less useful but still accepted by the system. [7] Too many and it becomes hard for your honest workers to enter their submissions properly, which could cause bad data or simply delays.
- For granularity, we can divide the tasks for workers into two types: simple and complex
  - Simple: Simple tasks are the ones that are (usually) short and self-contained. They generally are able to be solved by people of any qualifications, such as tagging or describing items. [8]
  - Complex: Complex tasks are ones that can be broken down into subtasks. For example writing a program, or a paper. These tasks usually require more experience or expertise than simple ones, which means it is harder to find people interested—or able—to complete these tasks. Often complex tasks are broken down into simple ones by the requester, then the simple tasks are crowdsourced.
- Picking proper incentives and compensation can affect the crowd's performance, on top of outcome quality. [9][10] We can categorize incentives and compensation policies into two parts: intrinsic and extrinsic.
  - Intrinsic compensation comes from inside the worker, and are things such as personal engagement or altruism.
  - Extrinsic compensation comes from outside the worker, and is typically monetary in nature.

  For monetary rewards, it is important to consider the amounts of payment, and also how they are structured. Higher monetary rewards tend to draw more workers, and decrease the time it takes said workers to complete the tasks. But increasing the monetary compensation doesn't necessarily equate to an increase in output quality. [11]

We will examine two types of approaches to designing a system that is easy to be quality controlled: design time and run-time.

- Creating the quality control system at design time is the first option. None of these methods will, of course, deter all malicious users or bad data, but they can do much in minimizing it. For example, defensive design is planning for malicious users to submit data, and making that submission of malicious data more challenging than doing the task properly would be. Another approach is properly selecting the workers. Common options are allowing anything to submit data, which pushes the quality control more towards the real-time, and post-processing, sections. Picking workers based on previous submission reputation, and selecting workers based on preverified credentials. See fig. 2 for more information on design time quality controlling.
- Controlling for quality at run-time is often used in conjunction with design-time approaches. Some effective, and popular, approaches are the expert review,

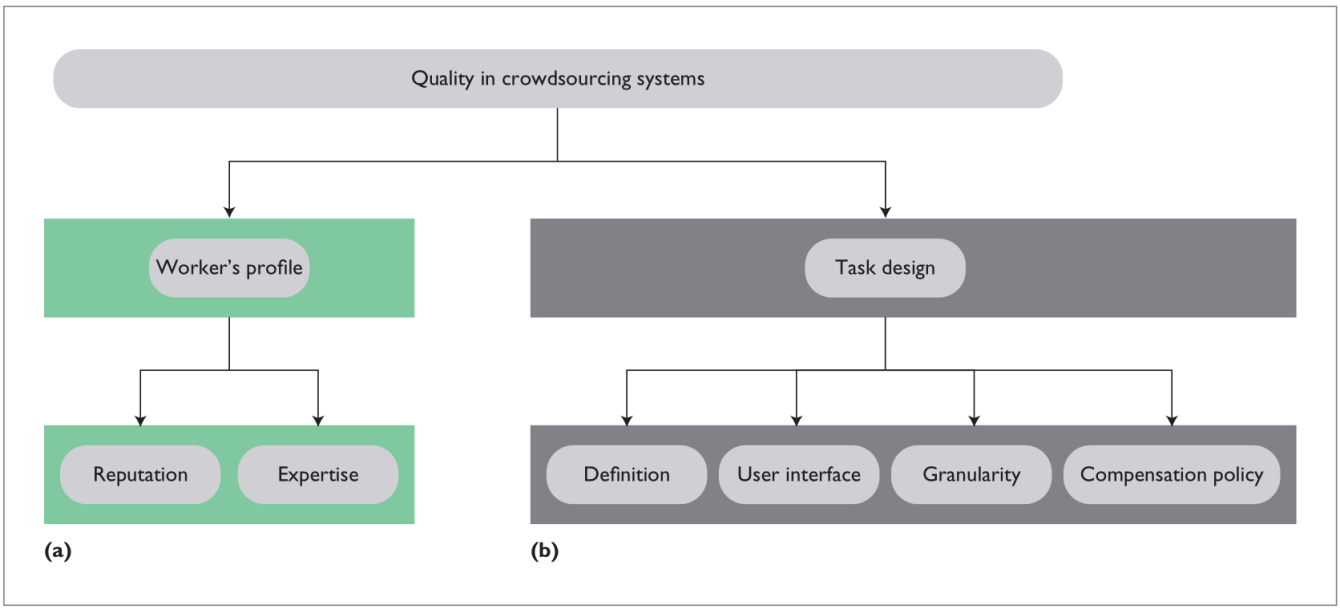

Figure 1: Taxonomy of quality in web-based crowdsourced systems.

or where experts in the field review the contribution before it is accepted. Wikipedia, especially, uses this approach. The methods we use for our formula, outlined later in this paper, will be contributor evaluation and a mix of output agreement and majority consensus. Contributor evaluation is when we assess the value of, and likelihood that, the contribution is useful and trustworthy based on prior submitted and rated data. This is only effective when there is prior data submitted by a user that can be analyzed. For our method we solve this problem by comparing submissions of users (output agreement), and that the submissions are similar enough to each other to be valid (majority consensus.)

2.2 Ubiquitous Crowdsourcing

Thanks to the rise of smartphones, a new type of crowdsourcing has been created. Ubiquitous crowdsourcing is smartphone owners contributing data about their outside world, such as GPS location, or ambient noise level. For example, Google Maps, or Waze, both rely on crowdsourced data to give information about traffic conditions, wrecks, and other events that happen on the road.

A major issue that faces developers who wish to use ubiquitous crowdsourcing is quality control. Just like in web-based crowdsourced, the data gathered from ubiquitous applications must be controlled for quality. For ubiquitous systems, quality control is even more important than web-based systems, in a lot of cases. On top of dealing with outright malicious users submitting bad data, the "fuzziness" of the real world requires us to deal with a truly huge amount of possible edge cases. Getting usable data out of the mess is what this paper will be discussing.

Two other issues that ubiquitous crowdsourcing faces that web-based crowdsourcing doesn't have to deal with are Real-time Events, and Dynamic Crowds.

- Real-time Events: ubiquitous crowdsourcing inherently deals with real-time events. The data is gathered in real time, and in many instances, real-time processing of the data is expected as well. In web-based crowdsourcing, on the other hand, quality control can often be delayed by some amount of time, if needed, to be checked and flagged by an authorized or more credible user.
- Dynamic Crowds: since ubiquitous crowdsourcing deals in real-time, the crowd itself is also often dynamic. People start and stop driving all the time, for example. The challenge of a dynamic crowd is that there are times when the number of contributors might not reach critical mass. Having too few points of data makes quality control even more important, on top of increasing the challenge of getting the proper results from the program.

Participatory sensing is the concept of communities (or other groups of people) contributing sensory information to form a body of knowledge. [12] A common problem with participatory sensing is data corruption, or malicious users intentionally sending invalid or fallacious data. The paper [13] shows a new concept of controlling for this by allowing consumers of the crowdsourced data to assign trust scores to specific sources of data.

The only other paper that is related to our work is. [14] In that paper, the authors consider the scenario where users deliberately try to confuse the sensor and send false data. The solution the authors propose is using a trust-based rating system, where each contributor has a reputation, or trust, rating and processed the data sent from users based on this rating. While their system shows an improvement over other trust-based systems, we propose an improvement to that system.

Our proposed method is to track the users' mobility patterns. Studies have shown that data can be very consistent when cross-referencing with a users' commute. For example, there is a high degree of regularity in a weekday commute;

Quality-control approach	Subcategories	Description	Sample application
Effective task preparation	Defensive design	Provides an unambiguous description of the task; task design is defensive—that is, cheating isn't easier than doing the task; defines evaluation and compensation criteria	
Worker selection	Open to all	Allows everybody to contribute to the task	ESP Game, Threadless.com
	Reputation-based	Lets only workers with prespecified reputation levels contribute to the task	Mturk, StackOverflow
	Credential-based	Allows only workers with prespecified credentials to do the task	Wikipedia, StackOverflow

Figure 2: Existing quality-control design-time approaches.

Quality-control approach	Description	Sample application
Expert review	Domain experts check contribution quality.	Academic conferences and journals, Wikipedia
Output agreement	If workers independently and simultaneously provide the same description for an input, they are deemed correct.	ESP Game
Input agreement	Independent workers receive an input and describe it to each other. If they all decided that it's a same input, it's accepted as a quality answer.	Tag-A-Tune
Group truth	Compares answers with a gold standard, such as known answers or common sense facts to check the quality.	CrowdFlower, MTurk
Majority consensus	The judgment of a majority of reviewers on the contribution's quality is accepted as its real quality.	TurKit, Threadless.com, MTurk
Contributor evaluation	Assesses a contribution based on the contributor's quality.	Wikipedia, Stack Overflow, MTurk
Real-time support	Provides shepherding and support to workers in real time to help them increase contribution quality.	
Workflow management	Designs a suitable workflow for a complex task; workflow is monitored to control quality, cost, and so on, on the fly.	

Figure 3: Existing quality-control run-time approaches.

most people tend to make a schedule and stick to it when traveling. [15] We look at taking a public bus, and users' traveling patterns to determine when the busses will arrive at given stops. First we define, for each user, Points of Interest (POIs) with tuples, $T(loc_{POI_x}, t_i)$. loc_{POI} is the specific point where the contribution was submitted. t_i is a logical time as opposed to a timestamp. We define a logical time as something application-relevant, such as *morning*, *afternoon*, or *late night*. While processing the data, we consider patterns differently for weekdays vs. weekends, since people generally have more regular schedules on the weekdays.

Based on this, and with this framework, we define a regularity function $Reg(T_j)$ whose values are calculated based on location readings from the users GPS. While there is obviously no history data from the start of the logging process, after just a few days of logging data we can begin to put together a fairly accurate regularity function for the user. We start to define POIs for the user as *local*, *familiar*, and *stranger*.

On top of the regularity function, we also define a reputation score. When the user submits data, it is checked against other users submissions and also their own regularity function to determine a trustworthiness score, $Trust(u_i)$. For web-based crowdsourcing, this kind of trust function is usually created by reviews on the users' input from experts, but in ubiquitous crowdsourcing the rating must be different, as there is both too much data and not enough time for éxpertsto rate each user. On top of that, using the formula we created, it is still possible to use data from locations where the user-submitted data is very sparse. Based on the regularity and trust functions, we can compute a credibility weight for user u_i:

$$\text{credibility weight}(T_j) =$$

$$\alpha \cdot Reg(T_j) + (1 - \alpha) \cdot Trust(u_i)$$

α can be adjusted on the fly to give more weight to either the trust or regularity functions for the overall credibility weight. For example, if there is an area with a scarcity of submissions, which is a known challenge to crowdrouscing applications, [16] it might be advantageous to shift the weight to regularity. While these submissions might not be as trustworthy on average, it is still better to have *some* contributions rather than none at all.

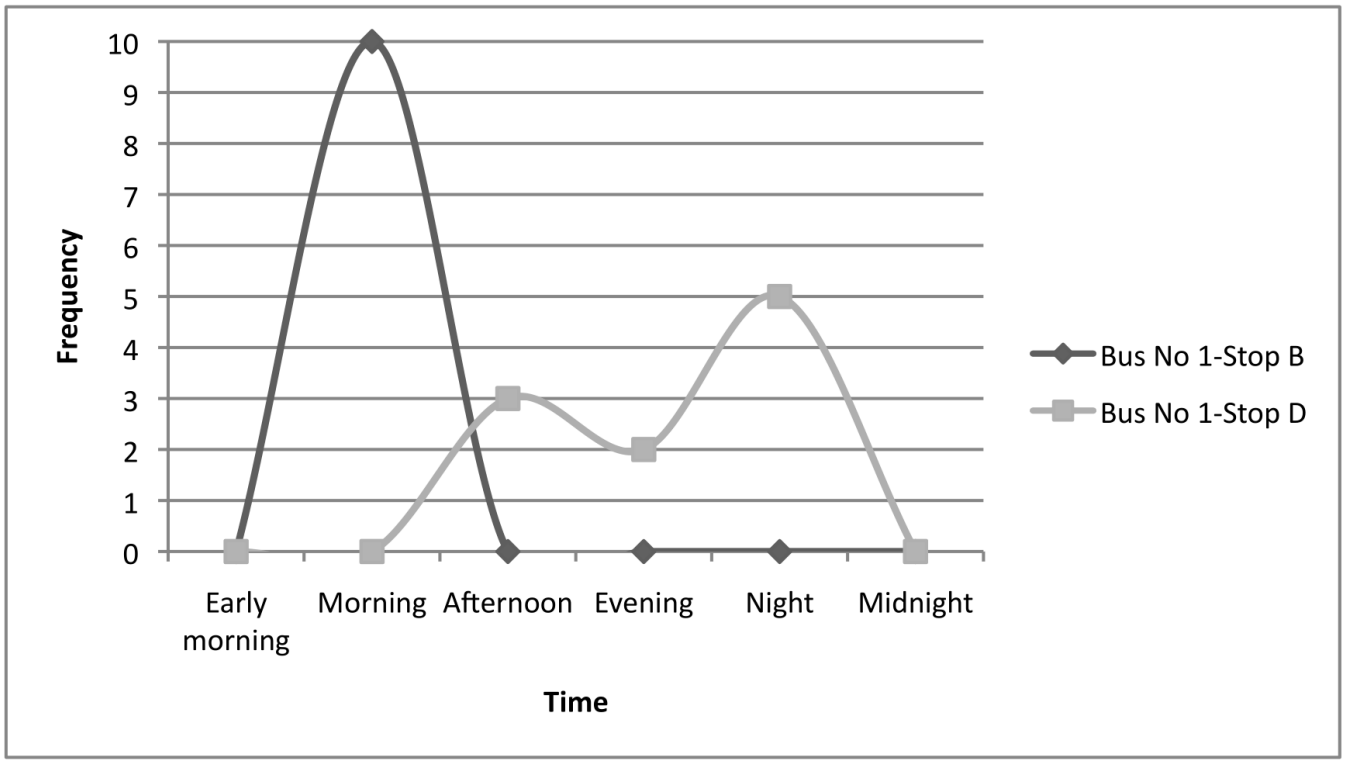

Figure 4: Frequency distributions for regularity function for a given user

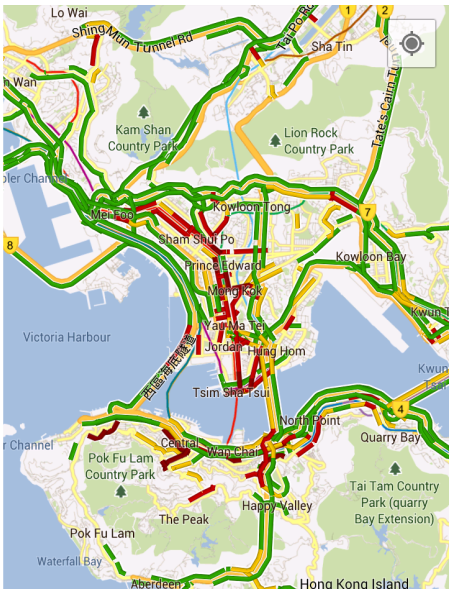

Figure 5: A popular use of ubiquitous crowdsourcing: Google Maps' traffic data

3. RELATED WORKS

Many websites are crowdsourced-based. The most popular is Wikipedia, the entirely user-submitted encyclopedia. Wikipedia is obviously not ubiquitous crowdsourcing, rather the first type of web-based crowdsourcing/data entry we mentioned, and so has different requirements for ensuring quality control than our formula, but it is still an excellent platform to look at. Many other such platforms exist, and often they use drastically different quality control methods. StackOverflow, for example, relies on a combination of explicit user ratings and also authorized moderators to police responses and questions. It is easy to see the design time strategies versus the runtime strategies the creators of StackOverflow have put into place.

4. CONCLUSION

We are creating an Android-based application that will give more accurate crowdsourced data about bus locations, based on the formula of this paper. We believe ubiquitous crowdsourcing has only just started to come into real force, and we want to show that its power is larger than was previously believed. Along with being powerful, we want to show the data is reliable, hopefully attracting many people to our application, giving more reliable data about bus positions.

We are paying special attention to the runtime factors of our quality control algorithms and designs. We hope our research encourages more people to develop crowdsourced applications and programs, allowing them to accurately and more easily be able to control for quality in their user submissions.

5. REFERENCES

[1] J. A. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava, "Participatory sensing," *Center for Embedded Network Sensing*, 2006.

[2] P. B. Crosby, *Quality is still free: making quality certain in uncertain times*. McGraw-Hill Companies, 1996.

[3] R. Khazankin, D. Schall, and S. Dustdar, "Predicting qos in scheduled crowdsourcing," in *Advanced Information Systems Engineering*, pp. 460–472, Springer, 2012.

[4] L. De Alfaro, A. Kulshreshtha, I. Pye, and B. T. Adler, "Reputation systems for open collaboration," *Communications of the ACM*, vol. 54, no. 8, pp. 81–87, 2011.

[5] D. Schall, F. Skopik, and S. Dustdar, "Expert discovery and interactions in mixed service-oriented systems," *IEEE Transactions on services computing*, vol. 5, no. 2, pp. 233–245, 2012.

[6] J. J. Chen, N. J. Menezes, A. D. Bradley, and T. North, "Opportunities for crowdsourcing research on amazon mechanical turk," *Interfaces*, vol. 5, no. 3, 2011.

[7] M. Allahbakhsh, B. Benatallah, A. Ignjatovic, H. R. Motahari-Nezhad, E. Bertino, and S. Dustdar, "Quality control in crowdsourcing systems: Issues and directions," *IEEE Internet Computing*, vol. 17, no. 2, pp. 76–81, 2013.

[8] A. Kittur, B. Smus, S. Khamkar, and R. E. Kraut, "Crowdforge: Crowdsourcing complex work," in *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pp. 43–52, ACM, 2011.

[9] O. Scekic, H. L. Truong, and S. Dustdar, "Modeling rewards and incentive mechanisms for social bpm.," in *BPM*, pp. 150–155, Springer, 2012.

[10] S. Dow, A. Kulkarni, S. Klemmer, and B. Hartmann, "Shepherding the crowd yields better work," in *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pp. 1013–1022, ACM, 2012.

[11] W. Mason and D. J. Watts, "Financial incentives and the performance of crowds," *ACM SigKDD Explorations Newsletter*, vol. 11, no. 2, pp. 100–108, 2010.

[12] B. E. H. R. R. S. S. W. Goldman, Shilton, "Participatory sensing, a citizen-powered approach to illuminating the patterns that shape our world," 2009.

[13] S. Reddy, V. Samanta, J. Burke, D. Estrin, M. Hansen, and M. Srivastava, "MobisenseâĂŤmobile network services for coordinated participatory sensing," in *Autonomous Decentralized Systems, 2009. ISADS'09. International Symposium on*, pp. 1–6, IEEE, 2009.

[14] K. L. Huang, S. S. Kanhere, and W. Hu, "Are you contributing trustworthy data?: the case for a reputation system in participatory sensing," in *Proceedings of the 13th ACM international conference on Modeling, analysis, and simulation of wireless and mobile systems*, pp. 14–22, ACM, 2010.

[15] I. Rhee, M. Shin, S. Hong, K. Lee, S. J. Kim, and S. Chong, "On the levy-walk nature of human mobility," *IEEE/ACM transactions on networking (TON)*, vol. 19, no. 3, pp. 630–643, 2011.

[16] A. Doan, R. Ramakrishnan, and A. Y. Halevy, "Crowdsourcing systems on the world-wide web," *Communications of the ACM*, vol. 54, no. 4, pp. 86–96, 2011.