Pertanika J. Sci. & Technol. 25 (S): 1 - 14 (2017)

SCIENCE & TECHNOLOGY

Journal homepage: http://www.pertanika.upm.edu.my/

Automated Update of Crowdsourced Data in Participatory Sensing: An Application for Crowdsourced Price Information

Fakhrul Syafiq*, Huzaifah Ismail, Hazleen Aris and Syakiruddin Yusof

College of Computer Science and Information Technology, Universiti Tenaga Nasional, Kajang, Malaysia

ABSTRACT

Widespread use of mobile devices has resulted in the creation of large amounts of data. An example of such data is the one obtained from the public (crowd) through open calls, known as crowdsourced data. More often than not, the collected data are later used for other purposes such as making predictions. Thus, it is important for crowdsourced data to be recent and accurate, and this means that frequent updating is necessary. One of the challenges in using crowdsourced data is the unpredictable incoming data rate. Therefore, manually updating the data at predetermined intervals is not practical. In this paper, the construction of an algorithm that automatically updates crowdsourced data based on the rate of incoming data is presented. The objective is to ensure that up-to-date and correct crowdsourced data are stored in the database at any point in time so that the information available is updated and accurate; hence, it is reliable. The algorithm was evaluated using a prototype development of a local price-watch information application, CrowdGrocr, in which the algorithm was embedded. The results showed that the algorithm was able to ensure up-to-date information with 94.9% accuracy.

Keywords: Automated algorithm, big data, crowdsourcing application, crowdsourced data, data deletion, data management, price information

INTRODUCTION

Crowdsourcing is "the act of taking a job traditionally performed by a designated agent (usually an employee) and outsourcing it to an undefined, generally large group of people in the form of an open call" (Howe, 2006). Today, 10 years after the term was coined, a large number of crowdsourcing initiatives exist in various forms and for various purposes. Crowdsourcing is used in news making (Alam & Campbell, 2012; Väätäjä,

ARTICLE INFO

Article history:
Received: 09 March 2017
Accepted: 18 September 2017

E-mail addresses:
mfakhrulsyafiq@gmail.com (Fakhrul Syafiq),
hujaepa@gmail.com (Huzaifah Ismail),
hazleen@uniten.edu.my (Hazleen Aris),
keisyakir@gmail.com (Syakiruddin Yusof)
*Corresponding Author

ISSN: 0128-7680

Vainio, & Sirkkunen, 2012), disaster relief management (Poblet, García-Cuesta, & Casanovas, 2014; Zook, Graham, Shelton, & Gorman, 2010), the creation of new products (Djelassi & Decoopman, 2013; Füller, Hutter, & Faullant, 2011) and fund raising (Althoff & Leskovec, 2015; Brabham, 2010), to name a few uses. Crowdsourcing is also used in information sharing (Aris & Din, 2014; Olleros, 2008). A more complete list of crowdsourcing examples can be found in the work of Aris and Din (2016, March), who list a total of 60 crowdsourcing initiatives. Leveraging on the widespread use of mobile devices, crowdsourcing initiatives are shifting to mobile platforms with the expectation of reaching larger crowds, leading to the birth of mobile crowdsourcing. Mobile crowdsourcing is a form of crowdsourcing that advertises and submits tasks through a mobile crowdsourcing application (MCA) installed in mobile devices (Väätäjä et al., 2012). Mobile crowdsourcing has led to two types of crowd involvement in crowdsourcing: participatory sensing and opportunistic sensing (Chatzimilioudis, Konstantinidis, Laoudias, & Zeinalipour-Yazti, 2012). In participatory sensing involvement, crowds participate through conscious contribution of data to the crowdsourcing initiative. The data are thus said to be manually generated by the crowd (Chatzimilioudis et al., 2012). Examples of MCAs that employ participatory sensing are Kpark (Davami & Sukthankar, 2015), Hazard Reporting (Chatfield & Brajawidagda, 2014), IndoorCrowd (Chen, Li, & Ren, 2014) and FloodPatrol (Yang et al., 2014). In these applications, information such as parking space availability and incidents of disastrous event comes from the users based on the knowledge obtained by observing things that are happening around them. The information is submitted using the applications through manual keying in of the information. On the other hand, in opportunistic sensing involvement, the crowd contribute data rather unconsciously because the data automatically 'sensed' by the sensors attached to the crowd's mobile devices (Chatzimilioudis et al., 2012). This is made possible by the technology available in most smartphones today that embed various sensors such as the global positioning system (GPS), gyroscope and accelerometer. Examples of MCAs that employ opportunistic sensing are PotHole (Simon, 2014), OpenSignal (OpenSignal, 2015), WeatherSignal ("About WeatherSignal," 2016) and VTrack (Thiagarajan et al., 2009). In these MCAs, data such as wireless signal strength and current temperature at a particular location are obtained through the sensors embedded in the mobile devices. Participatory and opportunistic sensing do not necessarily have to exist in mutual exclusivity in MCAs. Waze (2016), a widely known MCA, is an example of an MCA that employs both participatory and opportunistic sensing. Participatory sensing data are obtained from the response entered by the crowd, while opportunistic sensing data are obtained from the mobile device's accelerometer, which provides the speed of a moving car, and GPS, which provides its location. Waze thus automatically generates the traffic map based on GPS sensors and manual input from the crowd.

The use of mobile crowdsourcing in data collection poses new challenges due to the following nature of crowdsourced data that come mostly from volunteers.

1. Data that arrive at no specific time and are often at irregular intervals make it difficult to manually predict the frequency of incoming data.
2. Data that arrive in large volumes (big data) may be duplicated data, and therefore, redundant.

3. Data contributed by people from all walks of society, who come from different backgrounds and expertise, could mean that inaccurate, wrong or tampered data are possible.

In order for the crowdsourced data to remain updated, obsolete data have to be timely replaced with new data, or at least removed from the database even when without replacement. This is to ensure that only valid data are stored at any given time. Furthermore, due to the large volume of contributions to crowdsourced data, duplicate data are unavoidable. A selection or filtering mechanism is thus needed to decide on the data that should be captured and stored in the database. This is to ensure that the duplicate data do not overwhelm the database unnecessarily. Finally, to be useful and reliable, the up-to-date data captured from users have to be accurate too. This is one of the main challenges in crowdsourced data or big data (Agrawal, Das, & El Abbadi, 2011) i.e. inaccurate, wrong or tampered data. A mechanism that is able to detect the presence of invalid data is also needed.

To address the above challenges, an algorithm for automated update of these data through removal of obsolete and duplicate data is proposed. The objective is to ensure that at any particular point in time, application of the algorithm is able to ensure that only up-to-date and valid information is stored in the database. The detection and removal of duplicate and obsolete data have to be done automatically due to the nature of crowdsourced data, which come at an irregular and unpredictable rate, as mentioned in Point 1 above. The proposed algorithm is applicable to crowdsourced data with the following characteristics:

- Dynamic in nature, which means that frequent changes over a short period of time are expected; and
- Accuracy is important and slight deviation from accurate data may give a significant effect to render the information useless.

In this paper, the algorithm is presented and described. The algorithm focusses on participatory sensed data because in our opinion, this type of data is more prone to error introduced by the crowd. Furthermore, the scope of the algorithm is on data that are comparable, such as numerical data, which have exact accurate values rather than subjective data, which have a range of accurate values. For the purpose of demonstration and evaluation, the algorithm was implemented in a mobile crowdsourcing application, CrowdGrocr, which crowdsourced price information from users. The price information stored in the server (database) was compared with the actual price displayed at a number of selected stores under study to determine its accuracy.

Algorithm Construction

In this section, the construction of automated data removal of duplicate and obsolete dynamic crowdsourced data is explained. The algorithm comprises two parts. The first part is the detection and removal of duplicate data and the second is the detection and removal of obsolete data.

Removal of Duplicate Data

The first part of the algorithm targets two objectives: to prevent the large amount of data from swamping the database and to identify the correct data from the plethora of data entered. Both objectives are achieved by means of a series of simple steps embedded into the algorithm that determine the correct data by observing their frequency of occurrence. The assumption is that the higher the frequency of occurrence of the same data, the higher the likelihood that the data are correct. A minimum frequency, mf, is set and if the frequency of a series of consecutive duplicate data meets the mf value set, that data are deemed correct and hence, will be stored in the database. The first part of the algorithm is shown in Figure 1. As can be seen in Figure 1, the steps in the algorithm are triggered by data entry from the users i.e. the algorithm is automated. The algorithm begins in an idle state in which it is ready to receive data entered by the users. Each datum entered will increase the first counter, count1, which is initially set to zero, by one. The purpose of count1 is to identify if the data entered is the first time this data has been received. If it is, the data will be stored in the database. The reason for this is to ensure that as many data as possible are populated in the database. Correctness of data is dealt with later as more data are entered by the users. Each datum entered is also tagged according to the time entered, a *tstamp*. When a datum is selected to be stored in the database, the time entered, ts, which is taken from the *tstamp,* will also be stored.

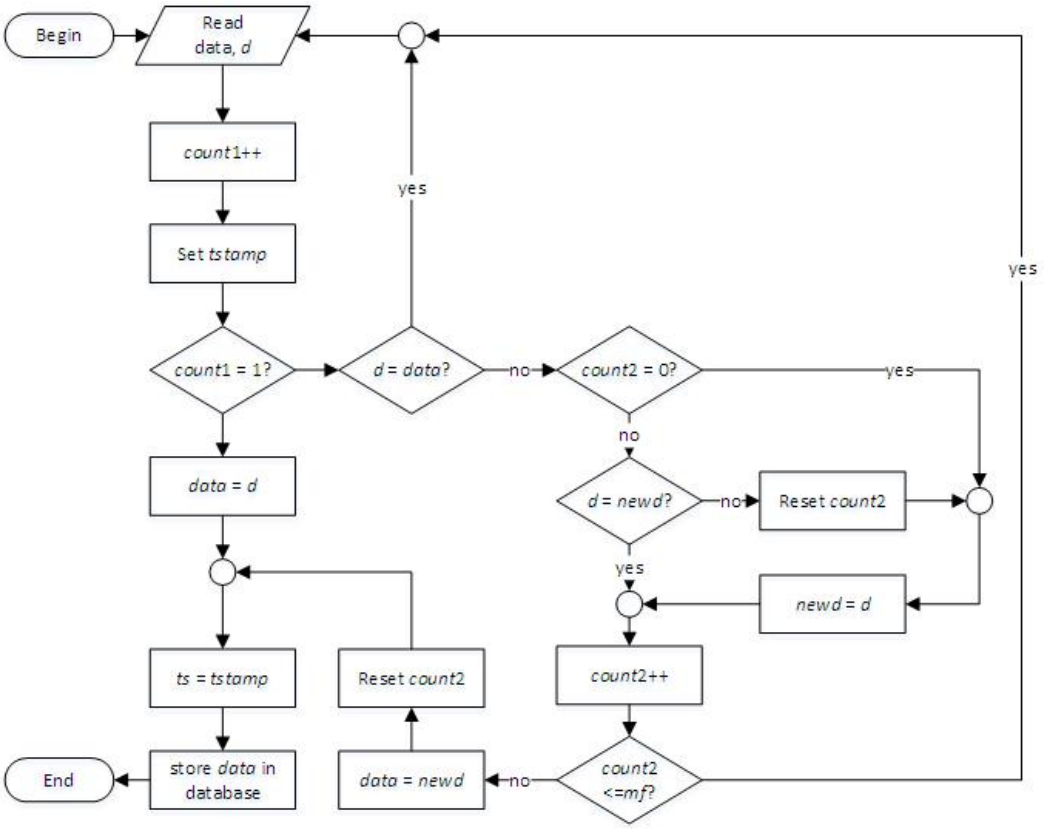


Figure 1. Automated detection and removal of duplicate data

If it is not the first time the data has been entered, the data will be compared with previously stored data in the database. If it matches previously stored data, meaning the same data have been received, the data will be discarded. If the data received are different from the value of the stored data, the difference is read as a potential change and the value of the second counter, count2, will be checked. If count2 is zero, the data will be temporarily stored as new data, newd, and the count2 value is increased by one point. If the count2 value is not zero, the data will be compared with the current value of newd. If they are the same, the count2 value will be increased further. If they are not, the count2 value is reset and the data will become the new newd value. After each increment, the count2 value is compared with the minimum frequency, mf, value set. If the mf value is reached, the data will be stored in the database together with its time stamp, ts, replacing the existing data. The count2 value will be subsequently reset as a result.

Removal of Obsolete Data

Dynamic crowdsourced data are subject to frequent changes, which means that the validity period of such data is comparatively short. Therefore, the stored data need to be refreshed from time to time to ensure that stored data are always up-to-date. Expired data need to be removed from the database, with or without replacement. For this reason, the second part of the algorithm focusses on detecting and removing expired data. Unlike the first part of the algorithm that is triggered by crowd input, the second part of the algorithm is designed to run continuously and repetitively at fixed time intervals, min, as shown in Figure 2. To begin with, the checking time, lastcheck, is set to current time, current. The algorithm will then run continuously, comparing the lastcheck with the current time. Each time the difference is larger than the predetermined time interval, min, all data in the database will be checked. The time stamp, ts, of each datum will be compared with the current time and if the difference (lapse) is larger than the defined duration, the data will be deleted from the database together with its associated time stamp. Subsequently, the count1 value will be reset to zero to indicate that the data are no longer available in the database and hence, the database is ready to accept new data as defined in the first part of the algorithm. At the end of the execution of the second part of the algorithm, the current time will be recorded as the new lastcheck. The next iteration of this part of the algorithm will begin when the difference between current time and lastcheck exceeds min and so on.

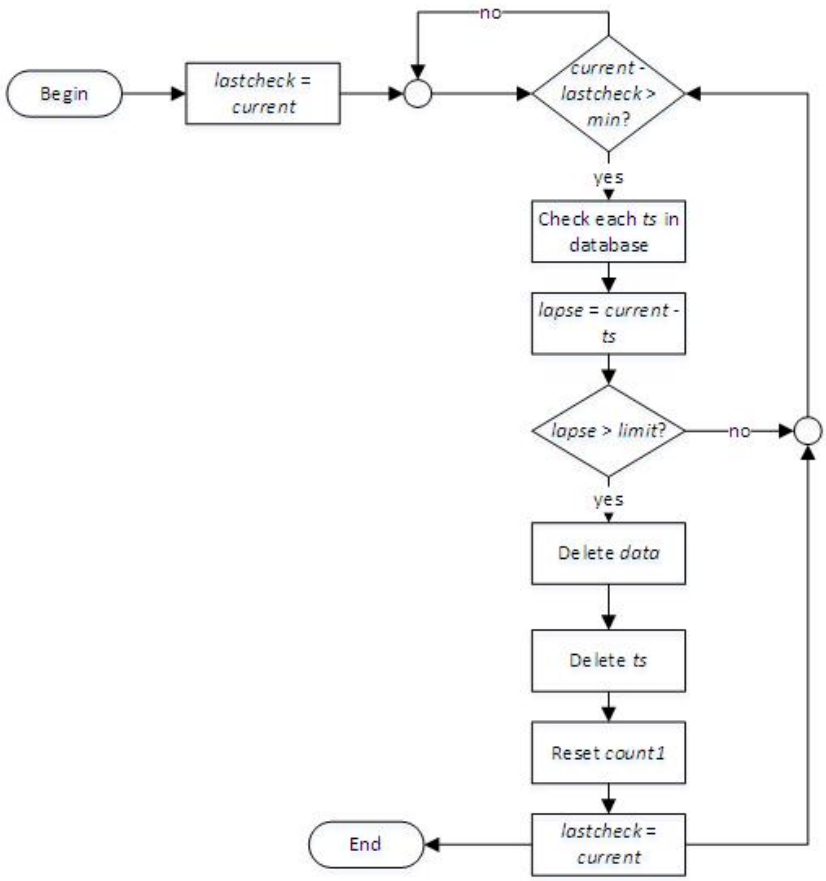


Figure 2. Automated detection and removal of obsolete data

Evaluation of Algorithm

For the purpose of evaluating the algorithm, a prototype of a local price-watch information application was developed, in which the algorithm was embedded. The prototype served as a means for users to enter information on item prices at store they visited. Therefore, the crowdsourced data in this case were the item prices at local supermarkets. Price information fits as a criterion of the crowdsourced data to be handled by this algorithm because price information changes from time to time i.e. it is dynamic, and each change matters. Changes in item prices can significantly affect household expenditure.

Instantiating the Algorithm

In order to apply the algorithm designed, three constants used in the algorithm need to be determined. These constants are mf, min and limit. To identify appropriate values for these constants in the context of price information, a series of surveys were conducted among a number of selected stores with the aim of obtaining information on the movement of item prices. Five stores were included in the survey; for confidential and privacy reasons, they

were called Store A, Store G, Store M, Store T and Store N. Brief profiles of these stores are shown in Table 1.

Table 1
Profiles of the selected stores

Store ID	Distance from Research Location (km)
Store M	11
Store N	11.4
Store G	13
Store A	9.4
Store T	5.1

The surveys were conducted on a weekly basis between 4 August 2015 and 26 August 2015. Initially, 20 grocery items were included in the survey; this was later increased to 26 due to inconsistent availability of some selected items at certain stores. At the end of the surveys, the following observations were made with regard to the movement of item prices in the market.

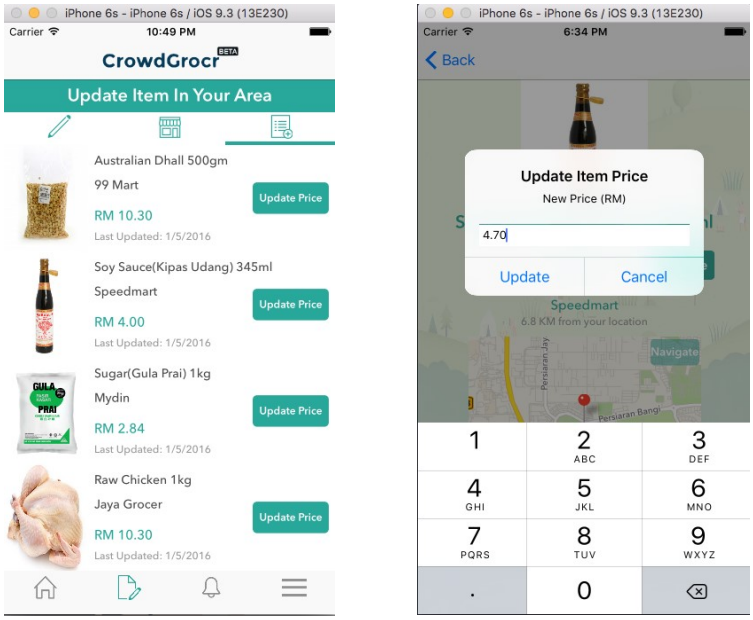
- Item price that changed only once a month: These items experienced price difference only once throughout the four weeks of the survey. Ten items in three stores were found to belong to this category.
- Item price that changed a few times during the month: This category of items experienced price changes at least two or three times during the four weeks of the survey. Five items at two stores belonged to this category.
- Price change during festive seasons or particular celebrations: At least two items were found to belong to this category, and they were detected based on a significantly large price drop.

Based on the observations above, the limit value, which is used in the second part of the algorithm, was set to one week because there was evidence of price changes after a week. This highlights that the second part of the algorithm that runs continuously compares the current time and the ts value of each price stored in the database. When the difference comes after more than a week, the price information will be regarded as obsolete and hence, removed from the database. Checking frequency, min, is determined by a number of factors such as rate of incoming data and processor capacity. We recommend that the frequency be based on the optimum trade-off value between processor capacity and the minimum period of data reaching obsolescence; this may require further research to formulate. For the purpose of demonstrating the implementation of the algorithm, we set the min value to five minutes; this means that the second part of the algorithm will continuously check each price in the database every five minutes. Finally, the mf value needs to be set. The mf value determines the number of occurrences of repeated price information entered that need to be temporarily recorded before the price value is accepted and stored in the database. We recommend this value to be determined based on the incoming crowdsourced data rate, which might be different depending

on the types of data. In our experiment, the mf value was set at three for the reason explained in the next subsection.

Prototype Development

For the purpose of the field experiment, a local price-watch information application, CrowdGrocr, which is an MCA, was developed. CrowdGrocr is an application that collects information on household item prices from its users. It provides functions that enable users i.e. the crowd to share information on prices of items at the local stores that they visit. If an item was previously recorded in the database, users need to only update its price. Otherwise, users may enter information about the item to record it as a new item. The prototype was developed based on the local price-watch information solicitation and sharing model in Aris and Md Din (2014). Adhesive mobile crowdsourcing application features described in Aris (2015, January) i.e. geo-tagging and automatic update of prices were also implemented in the prototype; the latter resulted in the construction of the algorithm discussed in this paper. Price information collected from the crowd through CrowdGrocr is stored in the database in such a way that it can be used to compare prices of items offered by nearby stores as shown in Figure 3(a). Thus, CrowdGrocr enables users to conveniently compare prices of items offered by nearby stores without having to leave the comfort of their homes. As justified in Aris (2015), crowdsourcing price information from users helps to overcome the problem of out-of-date price information faced by similar applications that use conventional data collection methods, such as those that use a system administrator or appointed agents (Syafiq, 2016).



(a) (b)

Figure 3. (a) Price comparison interface, CrowdGrocr, and (b) Price record interface, CrowdGrocr

Field Experiment

Field experimentation to evaluate the algorithm was conducted over two weeks between 18 January 2016 and 31 January, 2016. For the purpose of evaluation, the CrowdGrocr prototype was installed in participants' mobile devices. Open recruitment of participants was made through online social media advertisement. For this evaluation, there were no specific requirements asked of the users who could participate because price information is related to shopping and shopping is done by people of all ages. As long as they had the means to access the prototype and they were in the vicinity of the stores under study, they could participate. A total of 20 participants were recruited through online advertisement. The participants were evenly distributed among the five selected stores in Table 1. Each of them was responsible for visiting the assigned store during the two-week period of evaluation and recording the prices of 10 items in the given list in CrowdGrocr during the visit. The 10 items were selected based on the previous surveys conducted, from which it was found that some items were available in all the selected stores. Figure 3(b) shows the interface of the prototype. Apart from being asked to use CrowdGrocr, the participants were also asked to fill in the price information on paper forms as backup. To determine the accuracy of the algorithm i.e. its ability to ensure that up-to-date and correct price information was stored in the database at any point in time, manual checking of the prices by item was also performed by one of the researchers. The five stores were visited weekly during the two-week evaluation period and the prices of the same 10 items were recorded, together with the date and time of the visit. As the price of each item was manually checked, the researcher ran CrowdGrocr to check the price of the same item captured at that particular time.

RESULTS AND DISCUSSIONS

At the end of the two-week evaluation period, only 18 participants managed to submit the price information as required. A total of 98 manually checked prices of 10 selected items at the five stores mentioned above were collected, short of two prices because one item was not available at one of the stores. This represented 49 prices checked each week. Table 2 shows the comparison results of the 10 items in the first week of evaluation. Results of the comparison made between the manually checked prices and the corresponding prices shown by the CrowdGrocr application showed that 44 out of 49 prices were the same i.e. 89.8% accuracy was attained in the first week of the evaluation period. The results for the second week of the evaluation period showed improved accuracy, with 47 out of the 49 manually-checked prices being the same as the ones in CrowdGrocr, accounting for 95.9% accuracy. However, the table providing the evaluation results for the second week of evaluation is not shown here due to lack of space. Average accuracy was 94.9%.

Table 2
Comparison between price computed by the algorithm and the actual price displayed at supermarkets in the first week of evaluation

Item (Brand)	Store ID	Date & time	Actual Price (RM)	Computed Price (RM)
Chicken eggs Omega 3 (NUTRIPLUS) (10 pcs)	Store A	2016-01-23 16:40	5.45	5.45
	Store G	2016-01-23 17:17	5.39	5.39
	Store M	2016-01-23 14:36	5.29	5.29
	Store T	2016-01-23 15:40	5.19	5.19
	Store N	2016-01-23 18:21	4.60	4.60
Super special local rice (JATI) (10 kg)	Store A	2016-01-23 16:56	25.90	25.90
	Store G	2016-01-23 17:17	25.99	25.99
	Store M	2016-01-23 14:36	25.95	25.95
	Store T	2016-01-23 15:40	25.99	25.99
	Store N	2016-01-23 18:21	-	-
Body Wash Refill Pack (LIFEBUOY) (900 ml)	Store A	2016-01-23 16:37	14.30	10.88
	Store G	2016-01-23 17:14	14.14	14.14
	Store M	2016-01-23 14:33	13.37	13.37
	Store T	2016-01-23 15:38	10.88	10.88
	Store N	2016-01-23 18:18	11.30	11.30
Kicap manis (KIPAS UDANG) (345 ml)	Store A	2016-01-23 16:43	5.95	5.95
	Store G	2016-01-23 17:19	6.03	6.03
	Store M	2016-01-23 14:40	4.95	4.95
	Store T	2016-01-23 15:41	5.99	5.38
	Store N	2016-01-23 18:25	5.25	5.25
Milo Refill Pack (NESTLE) (2 kg)	Store A	2016-01-24 14:07	38.90	38.90
	Store G	2016-01-24 14:03	36.89	32.98
	Store M	2016-01-24 14:00	33.90	33.90
	Store T	2016-01-24 14:11	35.99	30.98
	Store N	2016-01-24 13:58	35.50	35.50
Crackers (JACOBS) (800 g)	Store A	2016-01-23 16:35	13.90	13.90
	Store G	2016-01-23 17:12	12.99	12.99
	Store M	2016-01-23 14:32	12.86	12.86
	Store T	2016-01-23 15:37	9.98	9.98
	Store N	2016-01-23 18:14	11.99	11.99
Sardine in tomato sauce (AYAM BRAND)	Store A	2016-01-23 16:48	4.05	4.05
	Store G	2016-01-23 17:25	3.99	3.99
	Store M	2016-01-23 14:44	3.99	3.90
	Store T	2016-01-23 15:47	4.15	4.15
	Store N	2016-01-23 18:30	3.99	3.99

Table 2 *(continue)*

Shampoo assorted (PANTENE) (350 ml)	Store A	2016-01-23 14:47	14.90	14.90
	Store G	2016-01-23 17:29	15.08	15.08
	Store M	2016-01-23 14:47	10.90	10.90
	Store T	2016-01-23 15:49	10.88	10.88
	Store N	2016-01-23 18:32	9.85	9.85
Chili sauce (LIFE) (500 g)	Store A	2016-01-23 16:54	3.65	3.65
	Store G	2016-01-23 17:32	3.81	3.81
	Store M	2016-01-23 14:49	3.40	3.40
	Store T	2016-01-23 15:52	3.59	3.59
	Store N	2016-01-23 18:35	3.55	3.55
UHT milk (DUTCH LADY) (1 L)	Store A	2016-01-23 16:56	5.95	5.95
	Store G	2016-01-23 17:34	4.98	4.98
	Store M	2016-01-23 14:51	4.99	4.99
	Store T	2016-01-23 15:55	4.99	4.99
	Store N	2016-01-23 18:37	5.50	5.50

Different prices are shown in bold in the table. This accuracy was achieved based on an average of six prices entered each week for each item at each store, hence the reason for setting mf at 3 previously i.e. at half of the expected frequency of incoming data. It is expected that a higher accuracy rate can be attained with a higher incoming rate of crowdsourced data. Apart from assessing the accuracy of the algorithm, the evaluation also obtained the following findings, which we believe are useful for crowdsourced data generally and price information specifically.

- **Erroneous data.** Users can potentially enter wrong data by mistake i.e. human error can occur. At the moment, the algorithm does not have the mechanism to deal with erroneous data of this sort. For this mechanism to be implemented, more research needs to be done to identify the range of valid values for data. Crowdsourced values that do not fall within the valid range should be discarded.

- P**rice difference.** A significant price difference can be seen for the same item at different stores i.e. up to RM3.42 in the case of the Body Wash refill pack as shown in Table 3. This supports the finding presented in Aris and Md Din (2016), and further emphasises the need to compare item prices before buying them as proposed in Aris (2015).

Complete details of the crowdsourced price information are not included in this paper due to space constraints. For the complete version, interested readers are invited to refer to the uploaded data set in Syafiq (2016).

Threats to Validity

While as much effort as possible was made to ensure the validity of this research, there were a number of constraints beyond our control that may have affected the generalisability of the research outcome. Firstly, out of hundreds of types of items sold at various stores, the survey only included 26 items and out of the many established stores in operation, the survey was done at only five supermarkets. We were not be able to cover all the items available and stores in existence due to time constraints. Furthermore, while conducting the surveys at the stores, the availability of selected items was not consistent. Certain items were out of stock for more than two weeks. In this situation, the price recorded was based on the displayed price tag on the shelf, which might not have been the same price if the item had been in stock. In addition, price changes within the week could not be detected because the surveys were only done once a week. However, people do not usually shop every week, as was reported by Aris and Din (2016). Therefore, monitoring price change on a weekly basis was regarded as sufficient. Finally, the price information gathered from the surveys did not take into consideration that the displayed price at a store might be incorrect because a price checking facility was not available at all the stores.

CONCLUSION AND FUTURE WORK

The widespread use of mobile devices and applications has resulted in the availability of an overflow of information, known as big data, and this has opened up a lot of research opportunity in the fields of managing, analysing and storing of data. Among the challenges in managing big data is ensuring that the data is correct and always up-to-date, especially if the data is dynamic in nature. In this paper, an algorithm for the automated removal of duplicate and obsolete dynamic crowdsourced data was presented as an answer to this challenge. The algorithm was evaluated by embedding it in CrowdGrocr application that crowdsources price information from the crowd. The evaluation results showed that the algorithm was able to ensure that up-to-date and correct data was stored in the database with 94.9% accuracy. Future work could incorporate a mechanism to detect outliers i.e. data that are out of valid range and more parameters to determine the accuracy of the provided data, such as location information and sender's reputation value.

ACKNOWLEDGEMENT

Information presented in this paper constitutes a part of the research titled 'The Construction of a Price-watch Information Solicitation and Sharing Model for Timely Price Comparison of Household Products using Mobile Crowdsourcing (FRGS/1/2014/SS07/UNITEN/02/1) funded by the Ministry of Higher Education Malaysia. We would like to record our appreciation and thanks to those who participated in the field experiment during the evaluation phase of this study.

REFERENCES

About WeatherSignal. (2016). Retreived from https://opensignal.com/blog/2016/03/15/forget-crowdsourcing-plume-labs-tries-flock-sourcing-with-connected-pigeons/

Agrawal, D., Das, S., & El Abbadi, A. (2011, March). Big data and cloud computing: current state and future opportunities. In P*roceedings of the 14th International Conference on Extending Database Technology* (pp. 530-533). ACM.

Alam, S. L., & Campbell, J. (2012, January). Crowdsourcing motivations in a not-for-profit GLAM context: the Australian newspapers digitisation program. In *ACIS 2012: Location, location, location: Proceedings of the 23rd Australasian Conference on Information Systems 2012* (pp. 1-11). ACIS.

Althoff, T., & Leskovec, J. (2015, May). Donor retention in online crowdfunding communities: A case study of donorschoose. org. In *Proceedings of the 24th International Conference on World Wide Web* (pp. 34-44). International World Wide Web Conferences Steering Committee.

Aris, H. (2015, January). An architecture for adhesive mobile crowdsourcing application. In *Mobile Computing and Ubiquitous Networking (ICMU), 2015 Eighth International Conference on* (pp. 64-65). IEEE.

Aris, H. (2015). Local Pricewatch Information Solicitation and Sharing Model using Mobile Crowdsourcing. In *Advanced Computer and Communication Engineering Technology* (pp. 449-457). Springer, Cham.

Aris, H., & Din, M. M. (2014, August). On using mobile crowdsourcing for timely information solicitation and sharing of prices. In *Software Engineering and Applications (ICSOFT-EA), 2014 9th International Conference on* (pp. 518-523). IEEE.

Aris, H., & Din, M. M. (2016, March). Crowdsourcing evolution: Towards a taxonomy of crowdsourcing initiatives. In *Pervasive Computing and Communication Workshops (PerCom Workshops), 2016 IEEE International Conference on* (pp. 1-6). IEEE.

Aris, H., & Din, M. M. (2016). Exploring the Potential of Mobile Crowdsourcing in the Sharing of Information on Items Prices. *Internal Journal of Advanced Computer Science and Applications, 7*(4), 540-549.

Brabham, D. C. (2010). Moving the crowd at Threadless: Motivations for participation in a crowdsourcing application. *Information, Communication and Society, 13*(8), 1122-1145.

Chatfield, A. T., & Brajawidagda, U. (2014, January). Crowdsourcing hazardous weather reports from citizens via twittersphere under the short warning lead times of EF5 intensity tornado conditions. In *System sciences (HICSS), 2014 47th Hawaii international conference on* (pp. 2231-2241). IEEE.

Chatzimilioudis, G., Konstantinidis, A., Laoudias, C., & Zeinalipour-Yazti, D. (2012). Crowdsourcing with smartphones. *IEEE Internet Computing, 16*(5), 36-44.

Chen, S., Li, M., & Ren, K. (2014, April). The power of indoor crowd: Indoor 3D maps from the crowd. In *Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on* (pp. 217-218). IEEE.

Davami, E., & Sukthankar, G. (2015, May). Improving the performance of mobile phone crowdsourcing applications. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems* (pp. 145-153). International Foundation for Autonomous Agents and Multiagent Systems.

Djelassi, S., & Decoopman, I. (2013). Customers' participation in product development through crowdsourcing: Issues and implications. *Industrial Marketing Management, 42*(5), 683-692.

Füller, J., Hutter, K., & Faullant, R. (2011). Why co-creation experience matters? Creative experience and its impact on the quantity and quality of creative contributions. *R and D Management, 41*(3), 259-273.

Howe, J. (2006). The rise of crowdsourcing. *Wired Magazine, 14*(6), 1-4.

Olleros, F. X. (2008, January). Learning to trust the crowd: Some lessons from Wikipedia. In *e-Technologies, 2008 International MCETECH Conference on* (pp. 212-216). IEEE.

OpenSignal. (2015). *Understanding the world's wireless networks*. See https://opensignal.com/methodology

Poblet, M., García-Cuesta, E., & Casanovas, P. (2014). Crowdsourcing tools for disaster management: A review of platforms and methods. In *AI Approaches to the Complexity of Legal Systems* (pp. 261-274). Springer, Berlin, Heidelberg.

Simon, P. (2014). *Potholes and Big Data: Crowdsourcing our way to better government.* '.Wired.' See: http://www. wired. com/2014/03/potholes-big-data-crowdsourcing-waybetter-government.

Syafiq, F. (2016). *Automated purging algorithm for duplicate and obsolete price information.* See https://www.researchgate.net/project/Automated-Purging-Algorithm-for-Duplicate-and-Obsolete-Price-Information

Thiagarajan, A., Ravindranath, L., LaCurts, K., Madden, S., Balakrishnan, H., Toledo, S., & Eriksson, J. (2009, November). VTrack: accurate, energy-aware road traffic delay estimation using mobile phones. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems* (pp. 85-98). ACM.

Väätäjä, H., Vainio, T., & Sirkkunen, E. (2012, October). Location-based crowdsourcing of hyperlocal news: dimensions of participation preferences. In *Proceedings of the 17th ACM international conference on Supporting group work* (pp. 85-94). ACM.

Waze. (2016). *Waze.* Retrieved from https://www.waze.com/about

Yang, D., Zhang, D., Frank, K., Robertson, P., Jennings, E., Roddy, M., & Lichtenstern, M. (2014). Providing real-time assistance in disaster relief by leveraging crowdsourcing power. *Personal and Ubiquitous Computing, 18*(8), 2025-2034.

Zook, M., Graham, M., Shelton, T., & Gorman, S. (2010). Volunteered geographic information and crowdsourcing disaster relief: a case study of the Haitian earthquake. *World Medical and Health Policy, 2*(2), 7-33.