Answering Complex Location-Based Queries with Crowdsourcing

Karim Benouaret†, Raman Valliyur-Ramalingam‡, François Charoy‡

†*Inria Nancy – Grand Est*
54600 Villers-lès-Nancy, France
karim.benouaret@inria.fr
‡*LORIA/Inria and Université de Lorraine*
54600 Villers-lès-Nancy, France
{*raman.valliyur-ramalingam, françois.charoy*}*@loria.fr*

***Abstract*—Crowdsourcing platforms provide powerful means to execute queries that require some human knowledge, intelligence and experience instead of just automated machine computation, such as image recognition, data filtering and labeling. With the development of mobile devices and the rapid prevalence of smartphones that boosted mobile Internet access, location-based crowdsourcing is quickly becoming ubiquitous, enabling location-based queries assigned to and performed by humans. In sharp contrast of existing location-based crowdsourcing approaches that focus on simple queries, in this paper, we describe a crowdsourcing process model that supports queries including several crowd activities, and can be applied in a variety of location-based crowdsourcing scenarios. We also propose different strategies for managing this crowdsourcing process. Finally, we describe the architecture of our system, and present an experimental study conducted on pseudo-real dataset that evaluates the process outcomes depending on these execution strategies.**

***Keywords*-Crowdsourcing, Location-Based Queries, Query Transformation, Process Management**

I. Introduction

Over time the World Wide Web has evolved beyond just being a platform for retrieving information. It has become an essential mean for sharing user-generated information and a communication platform for many people.

Furthermore, crowdsourcing, also known as human computation or social computation, is an emerging computation paradigm. People use increasingly crowdsourcing platforms such as Amazon Mechanical Turk[1], oDesk[2] and CrowdFlower[3] to solve problems that require human intelligence, experience and assistance.

With the development of mobile devices, and the prevalence of smartphones that boosted mobile internet-access, people can provide various types of data such as photos, location, etc. any time and any where. Therefore, it appears interesting to request contribution from people utilizing facilities of their device together with their intelligence and knowledge for answering location-based queries.

Substantial research work has addressed the problem of answering queries with crowdsourcing from different perspectives and within various communities, including database [1], [2], [3], information retrieval [4], [5], [6], image processing [7], [8], [9] as well as geographic information systems [10], [11], [12]. These works are important and useful, but we claim that they cannot support the kind of queries we are considering. Especially, these approaches focus on answering simple queries such as "take a photo of a given suspect" and "check the opening hours of a given store". Instead, we propose a crowdsourcing approach to answer complex location-based queries, i.e., queries that require the combination of several atomic tasks.

To illustrate the problem of answering location-based queries with crowdsourcing, imagine that the mayor of Nancy – a middle sized city from the North-East of France – wants to get a view on the state of the roads of the city to schedule repairs. He asks for citizens' contribution and collaboration to help him to achieve that goal. Two main reasons motivate the need of citizens' participation. First, in this kind of scenarios, information provided by humans (citizens) is more relevant than that provided by sensors and computers. Second, it can be done on a lower budget since citizens' participation is not directly rewarded.

The mayor may ask the following query: "what roads need a repair in Nancy". This query is simple when you ask experts to do it with a precise specification. It is more tricky when you want ordinary citizens to help you. Specifically, if hundreds of roads are reported as damaged, are all these roads really need a repair? And more importantly, how to find those that need an immediate repair?

Fortunately, smartphones are becoming increasingly sophisticated with cameras and wireless connectivity. In addition, according to digitalbuzz[4], mobile Internet usage should take over desktop Internet usage by 2014. Thus, the query can be changed to: "provide photos of roads that need a repair in Nancy along with an assessment {not damaged,

[1]https://www.mturk.com/mturk/welcome
[2]https://www.odesk.com
[3]http://crowdflower.com

[4]http://www.digitalbuzzblog.com/2011-mobile-statistics-stats-facts-marketing-infographic

COLLABORATECOM 2013, October 20-23, Austin, United States

DOI 10.4108/icst.collaboratecom.2013.254104

damaged, very damaged}". This query can be answered in a straightforward way, i.e., each participant send one or more photos of damaged roads along with their assessment. However, with this procedure, we may receive a lot of photos with different assessment for the same locations: some people may make mistakes, while some others may lie so that the roads in their vicinity will be repaired first.

A natural option is to use CrowdForge [13], [14], which gives the user the possibility to manually translate her/his main query into different tasks, and then the results of these tasks are combined to form the final results. However, the user may not know enough how to translate her/his query into different tasks. Thus, she/he needs to go through several trial-run processes to satisfy her/his needs. Another option is to use Turkomatik [15], [16], which leverages the crowd to propose a transformation of the query into simple tasks. However, from a crowdsourcing vision, this system presents a complex set of tradeoffs involving delay and accuracy.

In this work, we propose to overcome this problem with a more sophisticated process: *(i)* ask some citizens to provide photos of roads that need a repair. We refer to this step as *data collection*; *(ii)* ask some other participants to select the most representative photo of the problem for each location since a lot of photos of the same location may be provided. We refer to this step as *data selection*; it ensures that for the following step the crowd will have the same view on the problem at hand and it allows removing irrelevant photos; and *(iii)* ask some people to assess each selected photo with the different assessments defined by the user. We refer to this step as *data assessment*.

Unfortunately, designing and managing this process execution would be very painful or costly for the mayor (or any other user), and most probably not easy to repeat. First, he has to transform his query into small tasks realizable by humans and post them to the crowd. Then, he needs to decide how and when to move from one step to another one, i.e., from data collection to data selection, or from data selection to data assessment. This has an impact on the accuracy of the results and on the execution time of the process. It is important to set up an effective location-based crowdsourcing framework that would *(i)* automatically transform queries into simple tasks realizable by humans; and *(ii)* collect, aggregate and cleanse the data and answers provided by humans, and then return the results to the user.

In this paper, we argue that the process of data collection, data selection and data assessment can support a lot of scenarios, and we propose a location-based crowdsourcing framework to cope with the above-mentioned challenges. Our main contributions are summarized as follows:

- We propose a crowdsourcing process model, which aims at transforming a complex location-based crowdsourced query into an executable process;
- We present different execution strategies to manage the process leading into different kind of outcomes;
- We describe the design and the architecture of our implemented location-based crowdsourcing system;
- We experimentally evaluate the process outcomes depending on the proposed execution strategies, using pseudo-real dataset.

The remainder of this paper is organized as follows: Section II reviews related work. In Section III, we present our location-based crowdsourcing process model, while in Section IV we describe different processing strategies to manage the process. Section V gives an overview on our implemented system. Then, Section VI presents our experimental evaluation. Finally, Section VII concludes the paper.

II. Related Work

The work related to our falls in three crowdsourcing categories: optimizing crowdsourcing, location-based crowdsourcing and executing crowdsourcing processes. Hereafter, we provide some salient related work.

A. Optimizing Crowdsourcing

For quality assurance, current approaches; e.g.,[2], [12], [9], propose to assign each task to multiple humans. It is typical to have k assignments per task, and then aggregate the results using majority voting. In our work, the data selection and data collection steps also rely on this kind of aggregation.

On the other hand, there has been some recent work that focus on designing fundamental strategies for optimizing the quality of the results. In [17], the authors focus on finding the maximum, i.e., the highest ranked item in a set, using the crowd. The authors first introduce the judgement problem: given a set of comparison results, which element has the maximum likelihood of being the maximum element, and the next vote problem: given a set of results, which future comparisons will be most effective. Then, they prove that the optimal solution to both problems is NP-hard, and provide efficient heuristics that work well in practice. Finding the maximum in a crowdsourcing environment has also been considered in [18]. The authors first present several families of parametrized max algorithms that take as input a set of items and as output an item from the set that is considered as the maximum. Then, they we propose strategies that select appropriate max algorithm parameters. Parameswaran et al. consider in [19] the problem of filtering data with humans, and develop deterministic and probabilistic strategies to optimize the expected cost and quality. The work presented in [3] provides an estimated accuracy for results based on workers' historical performances. The main objective is to minimize the cost by predicting how many questions must be asked in order to achieve a given quality. In [20], the authors study the problem of jury selection by utilizing crowdsourcing for decision-making tasks on micro-blog services. More specifically, the problem is to enrol a subset of a crowd under a limited budget, whose aggregated wisdom

via majority voting has the lowest probability of drawing a wrong answer. To solve this problem, the authors propose efficient algorithms and effective bounding techniques. However, these approaches assume that the probability that each human provides the right answer is available, but in our case this information is not known.

B. Location-Based Crowdsourcing

With the rapid increase in smartphone technology, location-based crowdsourcing is becoming more popular. In [10], the authors propose a prototype for location-based mobile crowdsourcing consisting of a Web and a mobile client. Through these clients, people of the crowd can search for tasks and submit solutions. Similarly, in [11], the authors design a framework for crowdsourcing location-based queries on the top of Twitter. The Framework employs Foursquare to find the appropriate people to answer the query. In [12] Kazemi and Shahabi introduce a taxonomy of spatial crowdsourcing and then propose three various techniques for assigning the maximum number of tasks to the crowd. They also introduce a taxonomy for spatial data. Unfortunately, these frameworks cannot support complex queries.

Given a query photo, taken from a given location l, and a set of photos of the same location l retrieved by a search engine, Yan et al. propose in [9] a system that asks humans to vote for each photo to check whether it represents the query photo or not. The authors then propose some real-time techniques to find the appropriate photo. This process may be useful from location-based crowdsourcing vision, but cannot support collecting and handling masses of data.

C. Executing Crowdsourcing Processes

Despite all the studies on crowdsourcing, only a few studies have focus on executing crowdsourcing processes. The work presented in [13], [14] gives the user the possibility to translate manually her/his main query into different small tasks, and then the results of these tasks are combined to form the final results using a MapReduce-based approach. Similarly, in [21], the authors propose a crowdsourcing language allowing the user to transform her/his main query into a set of activities and to define the order in which these activities will be executed. However, the user may not know enough how to translate her/his query into different tasks. Thus, these approaches require a lot of efforts from the user. In [15], [16], the authors propose to leverage the crowd to transform the query into a crowdsourcing process. The crowd first propose a set of simple tasks realizable by humans. Then, the crowd select a set of interesting task. Finally, the crowd replace the query with a set of simple tasks. However, from a crowdsourcing vision, this approach presents a complex set of tradeoffs involving delay and accuracy.

Table I: Summary of Notations

Symbol	Description
O	objects name
C	context
R, l_i	region, location
A, a_j	domain of the assessments, assessment
T_s, T_e	start time, end time
J_i, h_j	jury, human
k	number of jurors
P_i, p_{ij}	set of photos of location l_i, photo of location l_i
SV_i, AV_i	selection voting, assessment voting
sv_j, av_j	selection score, assessment score
p_i^*, a_i^*	the most representative photo of location l_i in context C, the assessment of the photo p_i^* within context C

While these approaches are manual, demanding a high number of crowd dependency for every execution step, in our approach we use the declarative query model, which aims to maximize automation, based on the data collection, data selection and data assessment process.

Bernstein et al. present in [22], Soylent, a word processing interface that enables to short, proofread of documents. The authors introduce the Find-Fix-Verify pattern, which splits tasks into a series of generation and review stages. However, this pattern is designated for word processing and cannot support collecting data.

On the other hand, some approaches; e.g., [23], [24], propose to incorporate crowdsourcing activities into business processes. The main idea is to combine business processes and crowdsourcing. But these approaches remain very sequential compared to the execution strategies we propose.

III. Location-Based Crowdsourcing Process Model

The first challenge is to transform a complex location-based query into an executable process. In this section, we first define the input model of our framework. Then we describe how to extract the data collection, data selection and data assessment tasks from this input. Finally, we propose an output model. For reference, Table I summarizes the frequently used notations.

A. Input Model

We consider a simplified formalization of query to give users the flexibility to pose their queries on one hand, and to ease achieving our contextualized goal on the other hand.

The input for our framework is a query $\mathcal{Q}$, which follows the format of $< O, C, R, A, T_s, T_e, S >$, where O describes the set of objects the user is looking for, C describes the context of O that the crowd has to consider in answering $\mathcal{Q}$, R stands for the region, e.g., a city, $A = \{a_1, a_2, \ldots, a_m\}$ comprises the domain of the assessments that can be attributed to O in the context of C, T_s and T_e are respectively the start and end time of the query execution, and S is a parameter to select an execution strategy (we present

different execution strategies in Section IV, namely, Buffer, Deadline and FIFO).

The query in our example can be represented as: <roads, need repair, Nancy, {not damaged, damaged, very damaged}, 07/01/2013 – 8:00, 07/03/2013 – 20:00, Deadline>.

B. Tasks Model

Given a query $\mathcal{Q}$:-$< O, C, R, A, T_s, T_e, S >$, we formally define the data collection, data selection and data assessment tasks as follows:

Data Collection Task: We use photos as a means to retrieve the set of objects O and define a data collection task $\mathcal{DCT}$ as triple $< O, C, R >$. It asks the crowd to take photos of O within context C, in region R.

In our example, a data collection task is represented as <roads, need repair, Nancy>. It asks citizens to take photos of road that need repair in Nancy.

As a lot of photos of the same objects in a given location $l_i \in R$, e.g., photos of the same roads, are expected, we need to filter these similar photos to select the most representative photo of location l_i, in the context C. This step, i.e., data selection, ensures that for the next step, i.e., data assessment, all participants will have the same view on the problem and it allows removing irrelevant photos. We use the term jury to denote a set of humans that can make decision either on data selection or on data assessment. Formally, a jury $J_i = \{h_1, h_2, \ldots, h_k\}$ is a set of k humans. Each human $h_j \in J_i$ is called a juror.

Data selection task: given a set $P_i = \{p_{i1}, p_{i2}, \ldots, p_{in_i}\}$ of photos, within context C, and taken from the same location l_i, we define a data selection task $\mathcal{DST}$ as a triple $< P_i, C, l_i >$. It requires a jury J_i to vote "yes" for photos that represent location l_i within context C, and "no" for the remaining ones.

The answer to a data selection task $\mathcal{DST}$:-$< P_i, C, l_i >$ is called selection voting. Formally a selection voting $SV_i = \{(p_{i1}, sv_1), (p_{i2}, sv_2), \ldots, (p_{in_i}, sv_{n_i})\}$ is a set of tuples, where sv_j is the selection score the photo p_{ij}; it represents the number of "yes" votes for the photo p_{ij}. The photos p_{ij} with the highest selection score sv_j, with $sv_j \geq \lceil \frac{k}{2} \rceil$, i.e., he majority of jurors have agreed that p_{ij} represents location l_i in the context C, is called the most representative photo of location l_i within context C, and it is denoted by p_i^*. Note that if P_i comprises only one photo p_{ij}, i.e., $P_i = \{p_{ij}\}$, then p_{ij} is the most representative photo of location l_i within context C if and only if $sv_j \geq \lceil \frac{k}{2} \rceil$, i.e., the majority of jurors vote "yes" for the photo p_{ij}. Thus, it is typical to have k odd, allowing majority voting. It is also worth to note that, if several photos have the best score, a random one, among them, will be selected as the most representative photo.

Once the most representative photo p_i^* of location l_i within context C is selected, we need to assess it in order to return the observation in location l_i to the user.

Figure 1: Output Model

Data assessment task: given the most representative photo p_i^* of location l_i within context C and a domain of assessments $A = \{a_1, a_2, \ldots, a_m\}$, we define a data selection task $\mathcal{DAT}$ as a triple $< p_i^*, C, A >$. It requires a jury J_i to assess the photo p_i^* with an assessment $a_j \in A$ in the context C.

The answer to a data assessment task $\mathcal{DAT}$:-$< p_i^*, C, A >$ is called assessment voting. Formally an assessment voting $AV_i = \{(a_1, av_1), (a_2, av_2), \ldots, (a_m, av_m)\}$ is a set of tuples, where av_j is the assessment score the assessment a_j; it represents the number of jurors who assessed p_i^* with a_j. The assessment a_j with the highest assessment score av_j is called the assessment of the photo p_i^*, and it is denoted by a_i^*. Similar to the case of data selection, if several assessment have the best score, a random one, among them, will be selected as the assessment of the photo p_i^*. It is also worth to note that we can compute a mean instead of the mode. However, it is not applicable in all scenarios since the labels can change with scenario and the mean would not be meaningful. For example if the assessment is {uneven pavement, road pothole, road crack}, we cannot compute the mean.

C. Output Model

The output of our framework, which comprises the result of the query $\mathcal{Q}$, is a set of triples $< l_i, p_i^*, a_i^* >$, i.e., a set of photos of O along with their locations and assessments. For easy and better viewing the output, one can then invoke a map-based result visualisation service. Fig. 1 shows a screenshot of the output model of our system.

IV. Query Processing Strategies

The second challenge in building our framework is managing and executing the different tasks, and detecting citizen errors or misbehaviour – recall that the probability that each

human provides the right answer is not available in our case. The natural option to handle the errors is to distribute each task to k participants and to aggregate the answers. Our framework follows this direction for data selection and data assessment tasks ($|J_i| = k$; see Section III-B) since we assume that we will always find participants to provide such answers. However, a data collection task is more critical since *(i)* it requires citizens to be on-site to take photos; and *(ii)* it has an influence on the data selection and data assessment tasks. Of course, we can wait for k photos in each location l_i, but, this may take a very long time for some critical scenario like crisis management, where results are required insistently. On the other hand, in some other scenarios, where a high quality is required, k photos may not be sufficient to get the right result. Therefore, the number of photo that we need for each location depends on the scenario as well as the duration of the query, i.e., T_s and T_e. Given a query $\mathcal{Q}$:-$< O, C, R, A, T_s, T_e, S >$, we present in this section three different strategies, namely Buffer, Deadline and FIFO, to handle with these issues.

A. Buffer Strategy

Algorithm 1 shows the idea of the Buffer strategy. The process starts at T_s (loop in line 2) with crowdsourcing the data collection task (line 4). Then the following process (loop in line 5) is repeated until T_e. Waits for k photos of O in each location l_i then distribute the selection task regarding location l_i (lines 6–10). When k jurors complete their voting, the wining photo p_i^* is selected as the most representative photo of O for location l_i in context C, then the assessment task regarding location l_i is asked (lines 11-18), and wait for k votes. Once the assessment voting is done, the winning assessment a_i^* is considered as the appropriate assessment for the photo p_i^* and then the triple $< l_i, p_i^*, a_i^* >$ is returned to the user (lines 19–26). The process proceeds in the same manner for each location until T_e (loop in line 5).

Algorithm 1: Buffer

Input: a query $\mathcal{Q}$; an integer k;
Output: a set of triples $< l_i, p_i^*, a_i^* >$;

```
1  begin
2      while currentTime < T_s do
3          wait();
4      crowdsourceDCT(O, C, R);
5      while currentTime < T_e do
6          case providing(h_j, p_ij, l_i)
7              put p_ij in the corresponding set P_i;
8              if |P_i| = k then
9                  stop photos from location l_i;
10                 crowdsourceDST(P_i, C, l_i);
11         case selectionVoting(h_j, P_i)
12             put h_j in J_i;
13             update SV_i;
14             if |J_i| = k then
15                 stop selection voting in P_i;
16                 select winning photo p_i^*;
17                 remove all jurors from J_i;
18                 crowdsourceDAT(p_i^*, C, A);
19         case assessmentVoting(h_j, p_i^*)
20             put h_j in J_i;
21             update AV_i;
22             if |J_i| = k then
23                 stop assessment voting for photo p_i^*;
24                 select the winning assessment a_i^*;
25                 remove all jurors from J_i;
26                 return < l_i, p_i^*, a_i^* >;
```

B. Deadline Strategy

The buffer strategy starts data selection only when it gets k photos for a given location l_i. If citizens do not provide k photos of some locations at T_e, we lose those results. This situation is expected in practice since some locations are more visited than others. To overcome this limitation, we propose the Deadline strategy. Algorithm 2 outlines the online processing of this strategy. The process starts at T_s (loop in line 2) with crowdsourcing the data collection task (line 4). Then the idea is to collect photos of O starting from T_s until a deadline d (loop in line 5) and to built buckets of photos – each bucket regroups photos of the same location l_i. At time d, a data collection task is requested for each location l_i (loop in line 9). The next steps are similar to the Buffer strategy. After the selection voting, the wining photo p_i^* is selected as the most representative photo of O for location l_i in context C, and the assessment task regarding location l_i is asked (lines 12–19). Once the assessment voting is completed, the winning assessment a_i^* is considered as the relevant assessment for the photo p_i^* and then the triple $< l_i, p_i^*, a_i^* >$ is returned to the user (lines 20–27). This two steps proceeds in the same manner until T_e (loop in line 11).

C. FIFO Strategy

To move from the data collection phase to the data selection phase for a given location l_i, the buffer and deadline strategies wait for k photos of l_i or for the deadline d, respectively. It might be interesting to consider an execution that would generate results more instantly and where the result size would increase gradually. We present in Algorithm 3 the FIFO strategy, which proceeds as follows. The process starts at T_s (loop in line 2) with crowdsourcing the data collection task (line 4). When receiving a photo p_{ij} of O from a given location l_i and there is not a voting regarding location l_i, FIFO immediately asks participants to vote for this photo

Algorithm 2: Deadline

Input: a query $\mathcal{Q}$; an integer k; a deadline d;
Output: a set of triples $< l_i, p_i^*, a_i^* >$;

```
1  begin
2      while currentTime < T_s do
3          wait();
4      crowdsourceDCT(O, C, R);
5      while currentTime < d do
6          case providing(h_j, p_ij, l_i)
7              put p_ij in the corresponding set P_i;
8      stop photos from all locations;
9      foreach location l_i do
10         crowdsourceDST(P_i, C, l_i);
11     while currentTime < T_e do
12         case selectionVoting(h_j, P_i)
13             put h_j in J_i;
14             update SV_i;
15             if |J_i| = k then
16                 stop selection voting in P_i;
17                 select winning photo p_i^*;
18                 remove all jurors from J_i;
19                 crowdsourceDAT(p_i^*, C, A);
20         case assessmentVoting(h_j, p_i^*)
21             put h_j in J_i;
22             update AV_i;
23             if |J_i| = k then
24                 stop assessment voting for photo p_i^*;
25                 select the winning assessment a_i^*;
26                 remove all jurors from J_i;
27                 return < l_i, p_i^*, a_i^* >;
```

to see if it really represents location l_i in the context C (lines 6–10), and wait for k answers. Once the selection voting is done, if the majority of votes are "yes", i.e., the selection score sv_j of the photo p_{ij} is at least $\lceil \frac{k}{2} \rceil$, we select p_{ij} as a representative photo p_i^* for location l_i and thus a data assessment task is requested for p_i^*; otherwise, another selection voting is asked for the next photo $p_{ij'}$ of location l_i (lines 11–24), and the same procedure is repeated. When the assessment voting is completed, the winning assessment a_i^* is considered as the relevant assessment for the photo p_i^* and then the user is provided with the triple $< l_i, p_i^*, a_i^* >$ (lines 25–32). This strategy proceeds in the same manner for each location until T_e (loop in line 5).

V. System Overview

In this section we outline the basic components of our system, their roles and how they interact with each other. Our system is implemented using Bonita Open Solution[5], BPM system. The architecture of the system is illustrated in Fig. 2. The system consists of the following main components: *Process Generator*, the *Process Engine*, the *Task Manager* and the *Result Visualizer*. The Storage Engine is external, and is accessed by our system at query time. The *Process Generator* receives the query from the user and transforms it into a processing plan, i. e., it generates the data collection, data selection and data assessment process.

The *Process Engine* takes the processing plan from the *Process Generator* and generates a sets of tasks for the

[5]http://fr.bonitasoft.com/

Algorithm 3: FIFO

Input: a query $\mathcal{Q}$; an integer k;
Output: a set of triples $< l_i, p_i^*, a_i^* >$;

```
1  begin
2      while currentTime < T_s do
3          wait();
4      crowdsourceDCT(O, C, R);
5      while currentTime < T_e do
6          case providing(h_j, p_ij, l_i)
7              if there is not a voting regarding l_i then
8                  crowdsourceDST({p_ij}, C, l_i);
9              else
10                 put p_ij in the corresponding set P_i;
11         case selectionVoting(h_j, {p_ij})
12             put h_j in J_i;
13             update SV_i;
14             if |J_i| = k then
15                 stop selection voting in {p_ij};
16                 if sv_j ≥ ⌈k/2⌉ then
17                     stop photos from l_i;
18                     the winning photo p_i^* is p_ij;
19                     remove all jurors from J_i;
20                     crowdsourceDAT(p_i^*, C, A);
21                 else
22                     if P_i is not empty then
23                         remove next photo p_ij' from P_i;
24                         crowdsourceDST({p_ij'}, C, l_i);
25         case assessmentVoting(h_j, p_i^*)
26             put h_j in J_i;
27             update AV_i;
28             if |J_i| = k then
29                 stop assessment voting for photo p_i^*;
30                 select the winning assessment a_i^*;
31                 remove all jurors from J_i;
32                 return < l_i, p_i^*, a_i^* >;
```

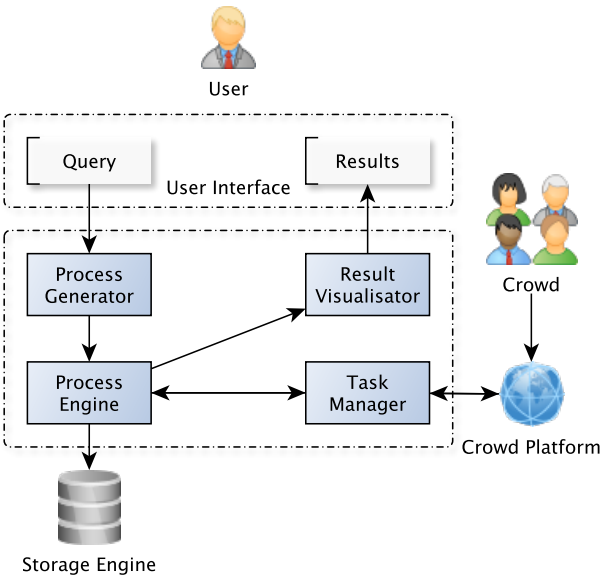


Figure 2: System Architecture

citizens to perform and aggregates the answers. Roughly speaking, the role of the *Process Engine* is to execute the process according to the strategy (Buffer, Deadline or FIFO) chosen by the user.

The *Task Manager* receives progressively the tasks from the *Process Engine* and communicates with the crowd via a crowd platform to post tasks and retrieve the answers, then send them to the *Process Engine*.

The role of the *Result Visualizer* is to receive the results from the *Process Engine* and to return them to the user as a map for easy and better viewing (see Fig. 1).

VI. Experimental Evaluation

The last challenge with our framework is to evaluate the quality of the results. Since we have not yet used it in a large scale setting, we propose to evaluate in this section the respective qualities of the proposed strategies, i.e., Buffer, Deadline and FIFO, focusing on: *(i)* the number of results returned; *(ii)* the quality of the results; for which, we use precision, recall and F-measure, i.e., $2\frac{precision.recall}{precision+recall}$; and *(iii)* the progressivity, i.e., how the results accumulate over time.

Due to the limited availability of large real datasets, most approaches focus on synthetic datasets. In our experimental study, we follow the direction of [12] and use the Gowalla dataset[6]. This dataset contains a set of users along with their check-in time and location on Gowalla, a location-based social networking website where users share their locations by checking-in. For our experiments, we assume that each user is a participant citizen. To generate the ground truth document, we make one random assignment of three possibilities for each location. We use a period of 5 days, and divide the data into three parts so that we have data collectors, data selectors and data assessors. We consider each check-in time in a given location as a response from that location for the current processing query. To be more realistic, for each collected data (photo), we introduce a parameter in $[0.5, 1]$ that controls whether the data selectors and data assessors provide a correct answer or not. That is, we assume that citizens have a high probability to provide correct answers than a wrong one.

[6]http://snap.stanford.edu/data/loc-gowalla.html

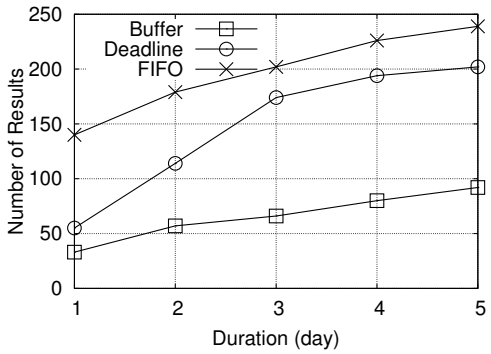


Figure 3: Number of Results vs Duration ($k = 7$)

For the deadline strategy, the deadline is set to the half of the duration of the process, i.e., $d = \frac{T_e - T_s}{2}$. For all strategies, the default values for the duration of the process, i.e., $T_e - T_s$ and the k value are 3 days and 7, respectively. Each experiment is run 100 times and the average it taken.

A. Number of Results

We measured the number of results, varying the duration of the process from 1 day to 5 days (Fig. 3). As expected, when the duration increases, the number of results increases since more citizens can participate. The number of results returned by Buffer is much less than that returned by Deadline and FIFO as Buffer needs to wait for k photos for each location, and some location are not popular. FIFO returns more results than Deadline since the latter strategy starts the data selection and assessment after the deadline, while the former, can do it whenever there is a photo received.

We also measured the number of results, varying k from 3 up to 11. Fig. 4 shows the results of this experiment. FIFO returns more results than the other strategies and remains unaffected since it does not wait for k photos or the deadline to start the data selection step. Deadline remains unaffected for $k \leq 7$. But for $k > 7$, the number of results decreases as the remaining time after the deadline d is insufficient to select and assess the retrieved data. Moreover, the number of results returned by Buffer decreases significantly with the increase of k since it requires k photos for each location.

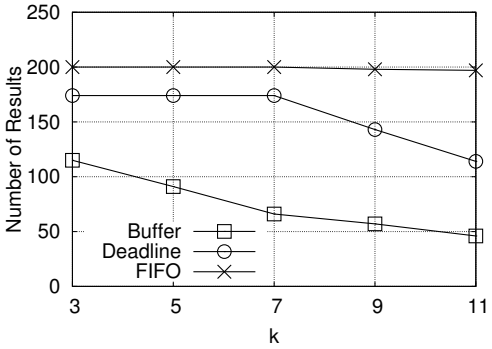


Figure 4: Number of Results vs k (Duration = 3 days)

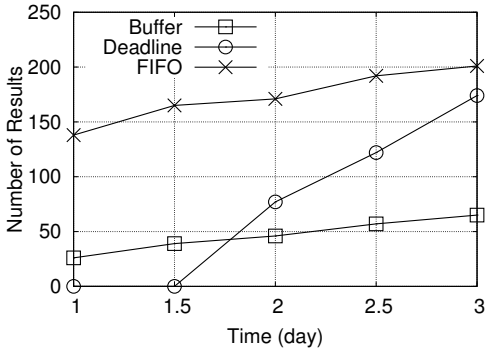


Figure 7: Progressivity (Duration = 3 days, $k = 7$)

To summarize, from Fig. 3 and Fig. 4, we obtain the following result:

> FIFO returns more results than Deadline, which in turn returns more results than Buffer.

B. Quality

Fig. 5 shows the quality of the results varying the duration. FIFO has the lowest quality, and is not affected by duration. In almost all cases, the first photo of each location is selected; since it is probably not the best photo of that location it affects the assessment step. The quality of Deadline and Buffer increases with the duration. Notice that Deadline is better than Buffer since it collects the maximum number of photos until the deadline d, and can move to data selection and assessment without having k photos for a location, while buffer, misses some location because some photos are missing to reach the threshold.

Fig. 6 depicts the quality of the results varying k. From this experiment, we can see that the quality of all strategies increases with the increase of k. When using more citizens or jurors, the probability to get a correct answer increases. Similarly to the last experiment, Deadline is better than Buffer, which in turn is better than FIFO for the same reasons.

Summarizing, from Fig. 5 and Fig. 6, we obtain the following result:

> Deadline has better quality than Buffer, which in turn has better quality than FIFO.

C. Progressivity

As shown in Fig. 7, FIFO and Buffer return some results the first day and progressively return the remaining ones. However, FIFO is around 3 times better since Buffer has to wait for k photos for each location. Moreover, the first results of Deadline can only appear after the deadline d, i.e., after 1.5 days, then increases significantly

From Fig. 7, we obtain the following result:

> FIFO is more progressive than Buffer, which in turn is more progressive than Deadline.

From our experimental study, we can observe that there is a very good tradeoff between the number, the quality and the velocity of occurrences of the results depending on the strategy we use:

> FIFO returns more results and is more timely, but less trustworthy than Deadline. Buffer is in between regarding quality and progressivity, but the worst regarding number of results.

VII. Conclusion

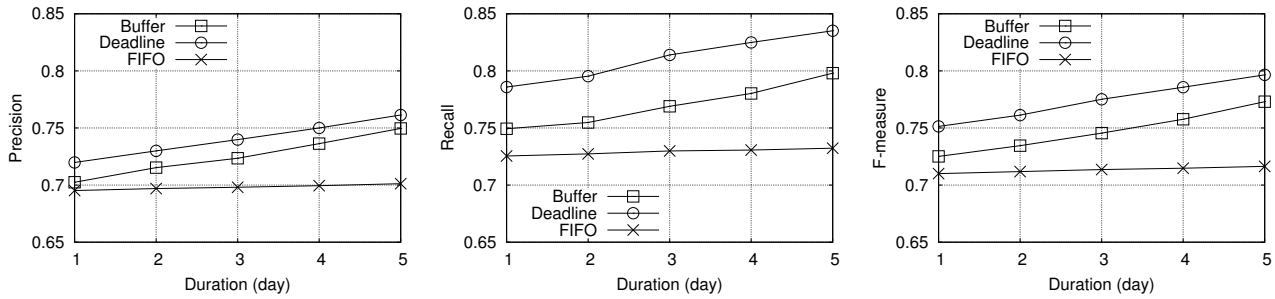
Our crowdsourcing framework leverages citizen participation to answer complex location-based queries in a given context. It is possible to follow different strategies based on data collection, data selection and data assessment process to answer these complex queries. Through our experimental evaluation, we see that each strategy has its merits. The choice of the strategy depends on the user needs. Hence, the proposed strategies are complementary. For example, if high accuracy is required, the Deadline strategy appears to be better but in a crisis situation where results are required urgently to take actions, the FIFO strategy is very well suited.

In the future, we plan to find the minimal set of strategies that cover all possible execution strategies and to conduct an experimentation on a large scale with the Lorraine Smart City Living Lab.

References

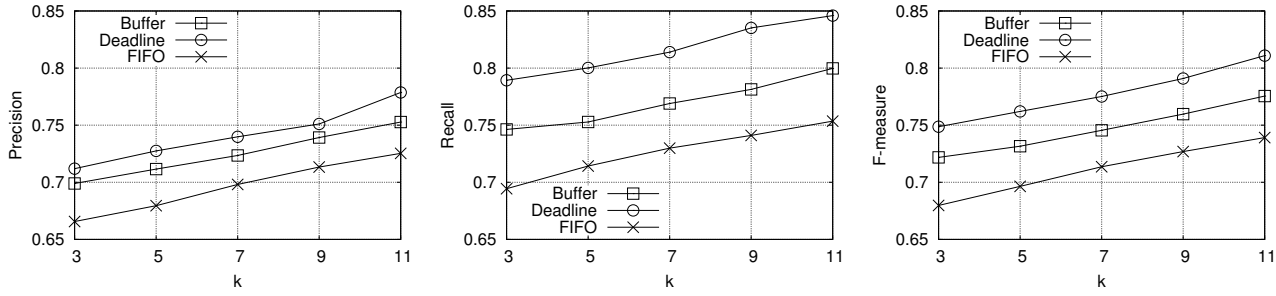
[1] A. Doan, M. J. Franklin, D. Kossmann, and T. Kraska, "Crowdsourcing applications and platforms: A data management perspective," *PVLDB*, vol. 4, no. 12, pp. 1508–1509, 2011.

[2] M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin, "Crowddb: answering queries with crowdsourcing," in *SIGMOD Conference*, 2011, pp. 61–72.



(a) Precision (b) Recall (c) F-measure

Figure 5: Quality vs Duration ($k = 7$)



(a) Precision (b) Recall (c) F-measure

Figure 6: Quality vs k (Duration = 3 days)

[3] X. Liu, M. Lu, B. C. Ooi, Y. Shen, S. Wu, and M. Zhang, "Cdas: A crowdsourcing data analytics system," *PVLDB*, vol. 5, no. 10, pp. 1040–1051, 2012.

[4] O. Alonso and M. Lease, "Crowdsourcing for information retrieval: principles, methods, and applications," in *SIGIR*, 2011, pp. 1299–1300.

[5] ——, "Crowdsourcing for information retrieval: principles, methods, and applications," in *SIGIR*, 2011, pp. 1299–1300.

[6] M. Lease and E. Yilmaz, "Crowdsourcing for information retrieval," *SIGIR Forum*, vol. 45, no. 2, pp. 66–75, 2011.

[7] K.-T. Chen, C.-C. Wu, Y.-C. Chang, and C.-L. Lei, "A crowdsourceable qoe evaluation framework for multimedia content," in *ACM Multimedia*, 2009, pp. 491–500.

[8] C.-L. Lee, Y.-M. Cheng, C.-L. Yeh, L.-C. Chen, W. Yu, and K.-T. Chen, "Surfing in the crowd: feasibility study of experience sharing in a taiwanese night market," in *OZCHI*, 2008, pp. 239–242.

[9] T. Yan, V. Kumar, and D. Ganesan, "Crowdsearch: exploiting crowds for accurate real-time image search on mobile phones," in *MobiSys*, 2010, pp. 77–90.

[10] F. Alt, A. S. Shirazi, A. Schmidt, U. Kramer, and Z. Nawaz, "Location-based crowdsourcing: extending crowdsourcing to the real world," in *NordiCHI*, 2010, pp. 13–22.

[11] M. F. Bulut, Y. S. Yilmaz, and M. Demirbas, "Crowdsourcing location-based queries," in *PerCom Workshops*, 2011, pp. 513–518.

[12] L. Kazemi and C. Shahabi, "Geocrowd: enabling query answering with spatial crowdsourcing," in *SIGSPATIAL/GIS*, 2012, pp. 189–198.

[13] A. Kittur, B. Smus, S. Khamkar, and R. E. Kraut, "Crowdforge: crowdsourcing complex work," in *UIST*, 2011, pp. 43–52.

[14] A. Kittur, B. Smus, and R. Kraut, "Crowdforge: crowdsourcing complex work," in *CHI Extended Abstracts*, 2011, pp. 1801–1806.

[15] A. P. Kulkarni, M. Can, and B. Hartmann, "Turkomatic: automatic recursive task and workflow design for mechanical turk," in *CHI Extended Abstracts*, 2011, pp. 2053–2058.

[16] ——, "Collaboratively crowdsourcing workflows with turkomatic," in *CSCW*, 2012, pp. 1003–1012.

[17] S. Guo, A. G. Parameswaran, and H. Garcia-Molina, "So who won?: dynamic max discovery with the crowd," in *SIGMOD Conference*, 2012, pp. 385–396.

[18] P. Venetis, H. Garcia-Molina, K. Huang, and N. Polyzotis, "Max algorithms in crowdsourcing environments," in *WWW*, 2012, pp. 989–998.

[19] A. G. Parameswaran, H. Garcia-Molina, H. Park, N. Polyzotis, A. Ramesh, and J. Widom, "Crowdscreen: algorithms for filtering data with humans," in *SIGMOD Conference*, 2012, pp. 361–372.

[20] C. C. Cao, J. She, Y. Tong, and L. Chen, "Whom to ask? jury selection for decision making tasks on micro-blog services," *PVLDB*, vol. 5, no. 11, pp. 1495–1506, 2012.

[21] P. Minder and A. Bernstein, "Crowdlang: A programming language for the systematic exploration of human computation systems," in *SocInfo*, 2012, pp. 124–137.

[22] M. S. Bernstein, G. Little, R. C. Miller, B. Hartmann, M. S. Ackerman, D. R. Karger, D. Crowell, and K. Panovich, "Soylent: a word processor with a crowd inside," in *UIST*, 2010, pp. 313–322.

[23] R. Khazankin, B. Satzger, and S. Dustdar, "Optimized execution of business processes on crowdsourcing platforms," in *CollaborateCom*, 2012, pp. 443–451.

[24] P. Kucherbaev, S. Tranquillini, F. Daniel, F. Casati, M. Marchese, M. Brambilla, and P. Fraternali, "Business processes for the crowd computer," in *Business Process Management Workshops*, 2012, pp. 256–267.