IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received February 18, 2019, accepted March 8, 2019, date of publication March 20, 2019, date of current version April 5, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2906506

A Dynamic Selection Approach for Quality Control Mechanisms in Crowdsourcing

REHAM ALABDULJABBAR[1] AND HMOOD AL-DOSSARI[2]
[1]Information Technology Department, College of Computer and Information Sciences, King Saud University, Riyadh 11451, Saudi Arabia
[2]Information Systems Department, College of Computer and Information Sciences, King Saud University, Riyadh 11451, Saudi Arabia

Corresponding author: Reham Alabduljabbar (ralabduljabbar@ksu.edu.sa)

This work was supported by the grant from the "Research Center of the Female Scientific and Medical Colleges," Deanship of Scientific Research, King Saud University.

ABSTRACT Crowdsourcing has emerged as a new model for leveraging human knowledge and intelligence toward accomplishing tasks that are difficult to fulfill effectively with machines alone. However, owing to its open nature, quality control is a big challenge. Current crowdsourcing systems use one or two standard mechanisms for evaluation and quality control of a task, regardless of its type. In this paper, we propose a dynamic approach that exploits task-quality ontology to select the most suitable quality control mechanism (QCM) for a given task based on its type. The proposed approach has been enriched by a reputation engine that collects requesters' feedback on the performance of QCMs. Accordingly, QCMs and tasks were automatically matched using the underlying categorization structure of tasks on one side and the reputation scores of QCMs on the other side. Our experiments establish that our proposed dynamic approach yields better results compared to existing approaches.

INDEX TERMS Classification, crowdsourcing, ontology, quality control, reputation, task.

I. INTRODUCTION

Technology has advanced to save human effort, time, and costs considerably to accomplish a task automatically. There are a number of tasks that require large-scale human involvement. For example, humans outperform machines in natural language processing and image understanding [1] that require basic human skills such as creativity and common sense [2]. According to [3], machines have not been able to match the complexity, creativity, and flexibility of human intelligence.

Crowdsourcing introduces a new way for organizations and individuals to leverage human knowledge and intelligence toward accomplishing special tasks that are difficult to fulfill effectively with machines alone. For example, the crowd may be invited to tag a photo, translate a written text, transcribe an audio file, or perform usability testing. However, due to the open nature of crowdsourcing, the quality control of the outcomes is a significant challenge.

Several studies have suggested a variety of mechanisms to relieve this concern [4]–[7]. Nevertheless, these mechanisms have mostly concentrated on managing the quality statically and have used the same quality control mechanism (QCM) for assessing different types of tasks. This approach has an inherent limitation in that a QCM that works properly for some tasks might be ineffective for others [8]. For instance, a QCM that is appropriate for assessing the outcome of language translation would not necessarily work well for assessing the outcome of image labeling because the two tasks differ from each other in their nature.

The nature of a task plays a vital role in crowdsourcing quality control [9]. Khasraghi and Aghaie [9] studied task-specific factors and found that the only factor that has a significant effect on the average quality of the accomplished tasks is the task type. This work is in line with a suggestion from [3], [8], and [10], who stated that a recommender system must be developed to maintain a dynamic quality model in crowdsourcing.

However, it is not easy for the requesters in crowdsourcing to identify the nature (i.e., type) of their required tasks. As highlighted in [8], it is difficult for the requesters to customize the quality control mechanism (QCM) based on their needs. Moreover, the requesters have little knowledge about the best QCM to direct the outcome of their submitted tasks. In addition, the sheer number of tasks would not help if these are unstructured.

In this paper, we propose a dynamic selection approach for the quality control of tasks in crowdsourcing. More particularly, we introduce a task ontology-based model that can

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Wang.

be utilized by a crowdsourcing system to determine the most appropriate QCM for a given task. The historical performance data of QCMs with different tasks collected over time enables to mine appropriateness and provide decisions for future tasks.

Broadly, the contributions of this paper are as follows: First, an overview of QCMs used in crowdsourcing systems is provided. Second, a dynamic selection approach for the quality control of tasks in crowdsourcing, consisting of three components, is outlined; (1) a task classifier to identify the types of crowdsourcing tasks automatically, (2) a Task_QCM ontology to standardize, share, and reuse the knowledge in the crowdsourcing domain, and (3) a reputation engine to collect and calculate QCM reputations. Third, an empirical evaluation of the classification component using real data obtained from Amazon Mechanical Turk (MTurk) [11], a well-known crowdsourcing system, is described. Finally, a simulation environment to evaluate the performance of the dynamic selection approach against existing quality control approaches is discussed.

The rest of the paper is structured as follows. Section II gives an overview of crowdsourcing, presents the state of the art in QCMs, and highlights important, relevant works. Section III formulates the research problem and provides details about the dynamic selection approach. In section IV, the evaluation methodology, details of the experiments, and results are presented. Lastly, sections V and VI discuss some challenges and provide conclusions, respectively.

II. LITERATURE REVIEW

A. CROWDSOURCING

The term *crowdsourcing* was coined jointly by the editors at *Wired Magazine*, named Howe and Robinson. It first appeared in Howe's article, ''The Rise of Crowdsourcing'', in 2006 [13]. The general principle of crowdsourcing is to distribute tasks to a crowd of workers who contribute their skills toward solving the problem. It has appeared under many different names, such as peer production, user-powered systems, collaborative systems, and community systems [14].

As is typical for an emerging area, several different definitions of crowdsourcing have appeared in the literature. Howe [15] defined crowdsourcing as ''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.'' Brabham [16], one of the academic pioneers in this field, defined crowdsourcing as ''an online, distributed problem-solving and production model.'' Estelles-Arolas and Gonzalez-Ladron-de-Guevara [17] studied and analyzed more than 40 definitions of crowdsourcing in the scientific and popular literature. They found that the most frequently cited definitions were those proposed by Howe [15] and Brabham [16].

Crowdsourcing has gained increased attention from both academia and industry [6]. It has been adopted by large online companies such as Amazon, Google, Facebook, Twitter, and Yahoo! [18] [19]. One of the most popular crowdsourcing systems for simple tasks is MTurk [11], and common systems for more complex tasks include Upwork [20] and Freelancer [21]. It is worth mentioning that MTurk refers to tasks as Human Intelligence Tasks (HITs), a term that has been used interchangeably with tasks in the literature.

Crowdsourcing has been recognized for its potential to enhance cooperation between computers and humans. It has been applied to handle real-world problems across different domains practically. For example, researchers at the University of Washington were not able to analyze the structure of protein for more than a decade. They posted the problem to the public in a crowdsourcing game involving 57,000 participants, and it was solved within three weeks [22].

Moreover, Markowsky [23] emphasized the vital role that the crowd plays in homeland security and the crime-fighting domain; for example, the crowd contributed to the identification of the perpetrators of the Boston Marathon bombing in 2013. In addition, Sabou *et al.* [24] highlighted the positive effects that the crowdsourcing has had on natural language processing.

Several crowdsourcing systems have emerged to serve specific application domains, such as CastingWords [25] that provides transcription services in a number of languages; and CrowdDB [26] that uses human input to process queries that neither a database system nor a search engine can adequately answer.

B. QUALITY CONTROL MECHANISMS IN CROWDSOURCING

Different mechanisms and techniques have been proposed to control the quality of crowd contributions [4]–[7]. These mechanisms can be employed at different levels in the crowdsourcing process (i.e., before posting the task) while the task is running, and after posting the task. Some of the most common QCMs are explained below.

1) QUALIFICATION TESTS

The aim of a quality control mechanism is to aid in the selection of workers before assigning a task to them. Several crowdsourcing systems provide requesters with the ability to develop their own qualification tests [27]. Such tests contain questions for workers that are similar to the real task. Only workers who pass these tests are allowed to work on the tasks. However, qualification tests do not actually evaluate the tasks after the solutions are submitted. They may only increase the possibility of getting better results; hence, other QCMs are needed alongside them for evaluating the tasks after the solutions are submitted.

2) GOLD STANDARDS

A very popular mechanism is the use of Gold Standard Tasks [6] to verify workers' performance. The main idea behind this mechanism is that tasks with previously known and defined solutions are inserted into the actual tasks that will be processed by the workers. If a worker's response to a Gold Standard Task matches the previously defined solution, his or her contribution to future tasks would be considered valid. This mechanism is used to infer the quality of workers' contributions and filter out poor-quality workers, and it has been incorporated into various crowdsourcing systems, such as CrowdFlower [28]. However, designing such tasks increases costs, and it can be difficult or even impossible in real-world situations to have such kinds of tasks [29].

3) REDUNDANCY

Some systems employ redundancy as their QCM to deduce the correct answer. Redundant work means allocating the same task to multiple workers [29]. Then, several statistical methods such as majority voting are applied to aggregate the responses [8]. However, such mechanisms typically work well for tasks where agreement among workers can be measured, such as with multiple choice questions. Therefore, they are not applicable for tasks with unstructured response formats, such as article writing or translation tasks, where agreement between workers is not expected.

4) WORKER REPUTATION

The reputation of workers is also commonly used as a metric for determining the quality of their responses, such as in [4], [5], and [30]. The reputation system reflects workers' task performance history. Moreover, spam-filtering techniques have been developed [31] to reduce noise in responses from malicious workers. However, current mechanisms calculate the worker's reputation in general without considering the type of task accomplished. That is, a worker might gain a high reputation by correctly accomplishing image-labeling tasks, but this does not necessarily imply that he or she can perform well in a text-translating task.

5) ITERATION-BASED

For more complex tasks with unstructured response formats, such as article writing or translation, an iterative process can be employed. In the iterative process, the solution of the first worker becomes an input for the next worker. Some systems use domain experts or peer review to check the quality of such tasks [32].

Researchers in the literature have proposed classifications for existing QCMs; for example, Allahbakhsh *et al.* [8] classified QCMs into design-time and run-time. Quinn and Bederson [32] described nine types of QCMs for human computation: output agreement, input agreement, economic models, defensive task designs, redundancy, statistical filtering, multi-level reviews, automatic checks, and reputation systems. In addition, Iren and Bilgen [33] identified the frequently used QCMs and grouped them according to their characteristics. And recently, Luz *et al.* [7] conducted an empirical study to examine the QCMs used in some of the existing systems. For instance, they pointed out that CrowdFlower relies on gold standards, and that CloudCrowd uses credentials and the credibility of workers as a quality indicator.

TABLE 1. Examples of QCMs applied to different tasks.

Task	Suggested QCM from the literature	Reference
Image labeling	Majority voting	[34]
Text translation	Tournament selection mechanism	[35]
Text annotation	Qualification test, then voting	[36]
Article writing	Iterative process	[37]
Image description	Input/ Output agreement	[38]
Text summarization	Find-Fix-Verify	[39]
Audio transcription	Iterative dual pathway	[40]

Among the mechanisms provided here, we are interested in proposing an approach that customizes the QCMs based on the nature or type of task. From the literature review, we can see that different QCMs are used to evaluate different types of tasks. Table 1 gives some examples of QCMs and the different types of tasks they can be applied to.

C. RELEVANT WORKS

As we have seen, many mechanisms have been proposed in the literature to handle quality in crowdsourcing; however, to the best of our knowledge, the issue of applying a QCM to tasks based on their nature has not yet been addressed. As such, we discuss below relevant research focusing on the ontology and task classifications in crowdsourcing systems.

In regard to ontology construction, the work of Hetmank [41] is relevant to our study. In this work, the author proposed a lightweight ontology for enterprise crowdsourcing. Yet, his research concerned with the development of an ontology for a special type of crowdsourcing task within an enterprise. Moreover, his ontology lacks regard for the quality control management in crowdsourcing systems, about which the author suggested more studies. We reuse and extend some parts of the enterprise crowdsourcing ontology and adapt it for our purposes. In addition, a skill ontology-based model for quality assurance was suggested in [30]. While this model was used to identify and match the best worker to a given task, our approach aims to identify and match the best-suited quality control mechanism to a given task. In our work, we argue that identifying the best worker is part of the whole QCM process.

Regarding task classification, another related study was conducted by Difallah *et al.* [42]. In their work, data collected from MTurk was analyzed with respect to some key dimensions of the system, including task type. They performed a large-scale classification of 2.5 million HIT types in order to study the evolution of HIT types on MTurk over time. Using the data labeled by MTurk workers, they trained a classifier for the task types proposed by Gadiraju [43]. Complementing such existing works with a different goal, our approach provides an ontological representation of tasks and QCMs as well as a classification of tasks as a basis for choosing a suitable QCM.

III. OUR PROPOSED APPROACH

In this section, we describe our proposed approach for QCM selection on crowdsourcing systems. The problem is formulated in section III-A, and then a detailed explanation of the proposed approach is presented in section III-B.

A. PROBLEM FORMULATION

It is important to first clarify what is meant by *quality* in the context of our work. In this paper, we adopt the definition of quality of a crowdsourcing task outcome as: "the extent to which the provided outcome fulfills the requirements of the requester" [8]. In other words, the quality of the task is dependent on the requester's satisfaction and whether or not the result has met his or her expected level of quality.

A typical scenario in crowdsourcing consists of a requester r who submits a task t to a set of Workers W who are registered in the crowdsourcing system and are willing to work on t and return the result to r. One essential goal for the crowdsourcing system is to implement task t and return the result to r at a satisfactory level of quality. However, as mentioned before, the current systems are limited to identifying a QCM that can control the task t. This paper attempts to address this problem. More specifically, given a set of tasks T, where $T = \{t_1, t_2, \ldots, t_n\}$, and a set of quality control mechanisms QCM, where $QCM = \{q_1, q_2, \ldots, q_m\}$, our objective in this paper is to select the best-suited QCM q to control and maintain the implementation of a given crowdsourcing task t.

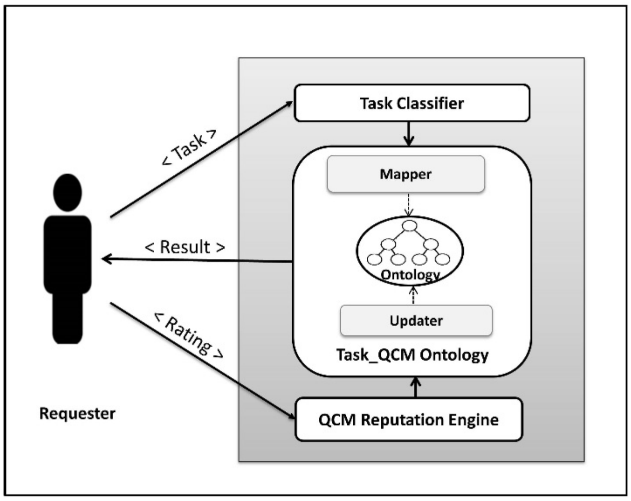


FIGURE 1. The process of the dynamic selection approach.

B. DYNAMIC SELECTION APPROACH FOR QCMs

The proposed dynamic selection approach consists of three components: Task Classifier, Task_QCM Ontology, and QCM Reputation Engine. Fig. 1 illustrates the process of the dynamic selection approach and the interactions between its main components. The role of each component is explained in detail in the following subsections.

1) TASK CLASSIFIER

Broadly, there are five types of crowdsourcing tasks: Opinion Based (OB), Content Generation (CG), Content Conversion (CC), Data Processing (DP), and Research Based (RB) [44]. A dynamic discovery of QCMs requires the provision of rich metadata for each requested task. The role of this component is to identify the type of the new incoming crowdsourcing tasks automatically based on their features.

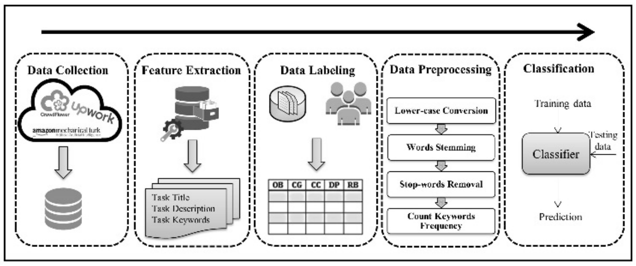


FIGURE 2. Classification component construction process.

Fig. 2 depicts the process of constructing the task classification component. The process starts by collecting real tasks from a crowdsourcing system. A task is usually represented by many features, such as task title, description, and keywords; hence, the feature extraction process aims to extract the relevant features for our work. After that, the data set is manually labeled based on the predefined task types. When the labeling process is finished, the labeled data set undergoes a preprocessing step that runs some filters for cleaning the data set. Finally, using the labeled data, a machine-learning algorithm is used for learning how to identify the task types of new tasks.

2) TASK_QCM ONTOLOGY

Ontologies are commonly used to facilitate knowledge representation, sharing and reusing. In crowdsourcing, task is the core element, and it can be classified into different types (e.g., image processing, video captioning, and audio transcription). In this section, we introduce the task ontology that can identify the most appropriate QCM for a given task [45].

The developed ontology acts as a conceptual backbone for selecting and matching QCMs to tasks based on their types. It was implemented using a well-known ontology editor called Protégé,[1] and stored using the descriptive language OWL (Web Ontology Language). To read data from these files, the Jena[2] library was used.

In general, there are two core concepts in the ontology: task and QCM. These concepts alongside their instances were acquired from an extensive literature study in crowdsourcing. Each task is described in the ontology by <*title*, *description*, *keywords*, *object type*, *action type*, *time allotted*, *reward*, *expiration date*, and *qualification required*>.

The *action type* means the action of the task; for example, tagging, describing, or categorizing. The *object type* means the data type of the object, such as text, image, or audio. Tasks in the ontology are grouped together based on their *task type,* which can be OB, CG, CC, DP, and RB, as described earlier. A *task type* is defined by the values of its set of common features. Two tasks with similar values of these

[1]http://protege.stanford.edu/
[2]https://jena.apache.org/

features are considered to be of the same *task type*. So, a task can be defined by its features; for example, the MTurk task *"Categorize the sentiment of a tweet as positive, negative, or neutral"* has the following features: [task type: OB, action type: categorization, object type: text].

Moreover, a profile was created in the ontology for each QCM to store its features, such as name, description, requirement, and other specifications. The performance of the QCM in evaluating the various tasks was stored in the ontology in order to calculate its reputation. The reputation score can be used for QCM selection and matching to a given task. It may be noted that the selection of a QCM for a given task can be applied if and only if the reputation score of the QCM exceeds a certain level otherwise, a random QCM is selected.

Once the crowdsourcing system receives a new request, the task classifier identifies the task type and passes it on to the Task_QCM ontology. The mapper then maps the ontology classes and their instances onto the tables in the knowledge base. Moreover, it is also responsible for gathering information using SPARQL queries over the knowledge base. In the next subsection, we explain how to select the most appropriate QCM for a given crowdsourcing task.

3) THE DynamicQCM ALGORITHM

The central challenge in our work was to select the most suitable QCM to evaluate and control the given task. For this purpose, we developed a DynamicQCM algorithm that basically selects the most reputable QCM based on its performance with previous similar tasks. Algorithm 1 describes how our proposed approach selects the most suitable QCM for the new crowdsourcing task.

Algorithm 1 The DynamicQCM Algorithm

```
1: Input: $t_n$ is a new crowdsourcing task,
          $O$ is the $TaskjQCM$ Ontology,
          $\alpha, \beta$ are given thresholds.
2: Output: $q$ is a quality control mechanism.
3: $max = 0$;
4: $k = 0$;
5: for each task $t_i \epsilon O$ do
6:     $sim = CalculateSimilarity(t_n{:}t_i)$;
7:     if $sim > max$ then
8:         $max = sim$;
9:         $k = i$;
10:    end if
11: end for
12: if $max \geq \alpha$ then
13:    if T-g $_t \geq \beta$ then
14:        Return $q_k$
15:    else
16:        Return $Random(q)$;
17:    end if
18: else
19:    Return $Random(q)$;
20: end if
```

The input of the algorithm (Line 1) is the required crowdsourcing task t_n; O, which is the Task_QCM ontology that holds the tasks, QCMs, and historical usage of QCMs with different tasks; α, which is a similarity score threshold that the tasks need to maintain; and β, which is the reputation score threshold that the QCMs need to maintain. The output (Line 2) is the selected QCM q to control and evaluate the new task t_n.

Two variables, *max* and k, are defined (Lines 3–4) to store the maximum similarity score between the new task $t_n$and other existing tasks, and the index of the task that has the maximum score (Lines 3 and 4), and their initial values are set to zero.

For each new task, the proposed algorithm calculates its similarity with other existing tasks (Lines 5–6). The algorithm then compares the value of *max* with the new similarity score that is stored in *sim* and updates the values of *max* and k if the new similarity score is greater than the value of *max* (Lines 7–10).

Finally, the algorithm checks whether the similarity score of *max* is greater than or equal to the value of α (Line 12); if so, then it checks whether the reputation score of the associated QCM with the most similar task exceeds or is equal to β(Line 13); if it does, then the associated QCM is selected for the control of the new task (Line 14); otherwise, if the similarity score or the reputation score of the QCM is less than the given thresholds, the proposed algorithm nominates a QCM randomly to control the new task (Lines 16–20).

To calculate the similarity between two tasks, the feature-based approach proposed in [46] was adopted in the DynamicQCM algorithm. This is because it fits our lightweight ontology structure, which depends on concept features more than hierarchy in presenting knowledge. In the similarity calculation, the relation between two tasks as dimensions of a vector space is modeled. That is, a Boolean scale is used to represent matching between the features of the new task and existing tasks. Similarity between tasks can then be measured using the set-based approach presented by Jaccard. However, we wanted to consider the importance of distinct features within the task. So, we assigned weights to features according to their importance in identifying similar tasks according to a survey conducted by Schnitzer *et al.* [48]. Similar tasks were then identified based on the Cosine distance [49] between their features. The similarity score was normalized to [0, 1], where 1 represents the most similar task and 0 represents the most non-similar task. The calculated similarity score was used to select the most suitable QCM for the new task. To explain how the proposed algorithm works, consider the following example.

Example: Suppose that the requester submits a new task t_{10}: *<DP, Classification, Audio>*, where *DP* is the task type, *Classification* is the action type, and *Audio* is the object type. Suppose also that the similarity scores betweent_{10} and existing tasks are calculated and presented as in Table 2.

In our calculation, the weights assigned to task type, action type, and object type were 0.6, 0.3, and 0.1, respectively.

TABLE 2. Similarity score calculation between task t10 and all existing tasks.

Task ID	Task type	Action type	Object type	QCM	QCM reputa tion	Cosine similarity score
t_1	OB	Classification	Image	q_2	0.89	0.44
t_2	DP	Classification	Image	q_1	0.87	0.98
t_3	DP	Describing	Image	q_8	0.90	0.88
t_4	CC	Writing	Text	q_3	0.88	0
t_5	CI	Describing	Image	q_6	0.85	0
t_6	CI	Translation	Text	q_5	0.81	0
t_7	CI	Summarization	Text	q_4	0.93	0
t_8	CI	Transcription	Text	q_9	0.27	0
t_9	CI	Transcription	Audio	q_7	0.80	0.15

These weights can be set and updated by the coordinator of the crowdsourcing system. For example, the similarity score between $t_{10}$and t_1 equals 0.44. This is because only one feature in t_1 (i.e., action type) has a similar value with the same feature in t_{10}. So, 0.44 represents the cosine similarity score between those vectors: u <0, 0.3, 0> for t_1 and v <0.6, 0.3, 0.1> for t_{10}.

It may be noted that the QCM column in Table 2 refers to the most reputable QCM for each given task. This information was computed by the reputation engine (as we will see in the next section) and retrieved from the Task_QCM ontology.

Now, let us assume that the similarity threshold (α in Algorithm 1) equals 0.5. In the best case, there would be a task that is exactly similar to the new task t_{10}. However, in our case, it is easy to see that only one task, t_2, would be considered the most similar to t_{10}. Therefore, the QCM associated with t_2 (q_1) would be selected to evaluate and control the execution of t_{10}. In some cases, more than one task would be similar to the new task. For example, suppose that the similarity score between t_{10} and t_3 is also 0.98; in this case, the most reputable QCM would be nominated and assigned to control t_{10} (the one associated with t_3, q_8). If, for example, the most similar task to t_{10} is t_8, the system would nominate a QCM randomly because t_8 is associated with a QCM that has a low reputation score (i.e., the associated QCM would be considered if its reputation score exceeds a given threshold, which is β in Algorithm 1, say 0.6). The random approach is adopted in order to give each QCM an equal opportunity of being selected with various tasks, and hence the reputation of each QCM would be built over time.

After the crowdsourcing system implements t_{10} using the selected QCM and returns the result to the requester, the requester would be asked to provide his or her rating. The provided ratings would be used by the QCM reputation engine to recalculate the reputation score of the used QCM, as we will see in the next section. The updater in the Task_QCM ontology would be notified to update the QCM and task association accordingly.

4) QCM REPUTATION ENGINE

This component is responsible for the calculation of a QCM's reputation. As we said earlier, the system holds historical performance data of QCMs. The data reflect the requesters' ratings on QCM performance when applied to different tasks. Requesters' ratings can be specified in terms of their satisfaction with the received result and are collected after receiving the final result from the system. The rating ranges from 0 to 1, where 0 indicates bad quality and 1 denotes excellent quality.

Based on the above aspects, a reputation score is established for each QCM associated with the requested task. Algorithm 2 describes the functioning of the proposed QCM reputation algorithm.

Algorithm 2 The QCM Reputation Algorithm

```
1: input: $t$ is a given crowdsourcing task
          $q$ is a quality control mechanism
          $D$ a historical usage of QCMs with different tasks.
2: output: $r$ a reputation of $q$ in controlling task $t$.
3: sum = 0;
4: count = 0;
5: for each data instance $d_i \in D$ do
6:     if $(qi=q)$ &$(t_i = t)$ then
7:         sum = sum + $r_i$;
8:         count = count + 1;
9:     end if
10: end for
11: $r = \frac{sum}{cosnt}$;
12: Return $r$;
```

The input parameters of the QCM reputation algorithm are the task t, the quality control mechanism q for which we need to calculate its reputation in controlling t, and the historical usage D of QCMs with different tasks (Line 1). The output is the calculated reputation score r (Line 2).

In the historical usage D data, each single instance has three values: $< t, q, r >$, where t is the task, q is the QCM, and r is the rating that represents the effectiveness of q in controlling t. The algorithm uses two variables: *sum* and *count*, the first for the rating summation (Line 3) and the second for calculating the number of ratings (Line 4). Subsequently, the algorithm scans the rating instances in D (from Line 5 to Line 10).

For each instance, the algorithm checks whether the task and quality control are equal to the target values (Line 6); if so, its rating would be considered in the calculation (Line 7 and 8). After the algorithm finishes scanning of all the existing instances, the reputation score is calculated by dividing the summation of the ratings into the total number of ratings (Line 11), and the result is sent to the Task_QCM ontology (Line 12). The updater in the Task_QCM ontology then reorders the list of QCMs using the new reputation score and updates the associated QCM with the targeted task, if necessary.

IV. EVALUATION AND RESULTS

Our evaluation consisted of two main parts. First, we evaluated the task classifier based on the real data collected from MTurk. Second, we conducted a set of experiments to compare the DynamicQCM approach against other quality control approaches using synthetic data generated from a simulation environment. This is because real experiments in such systems are not only time consuming but also difficult to conduct. The prototype was implemented using the Eclipse IDE, with the Java programming language.

A. EXPERIMENT 1: CLASSIFYING CROWDSOURCING TASKS

As we described previously, there are five types of crowdsourcing tasks: Opinion Based (OB), Content Generation (CG), Content Conversion (CC), Data Processing (DP), and Research Based (RB). In this experiment, we showed how a machine learning algorithm can be used to infer the type of a crowdsourced task automatically. We used the real data collected from MTurk [11], and classified them using the WEKA toolkit [50]. Three machine-learning algorithms were used to classify the crowdsourced tasks: Support Vector Machine (SVM), Naïve Bayes (NB), and k-Nearest Neighbor (k-NN). The component was evaluated with 10-fold cross validation over the labeled data.

1) DATA PREPARATION

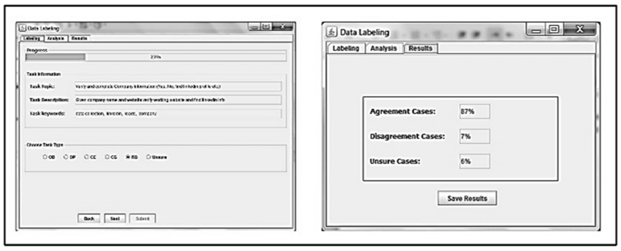
The data set used in this work was randomly collected from MTurk [11]. MTurk was chosen as it was one of the first crowdsourcing systems in the market, and it remains the most prominent system [3]. In August 2014, MTurk reported having more than 500,000 workers from 190 countries worldwide [51]. The MTurk-Tracker [52] shows that MTurk lists 150,000-300,000 tasks to be executed per day on an average. To obtain a wider data set sample spanning various types of tasks, we specifically sourced the data using MTurk-Tracker [52]. MTurk-Tracker has collected data about HITs published on MTurk periodically over the past five years. Their data is available at http://mturk-tracker.com/.

Overall, the obtained data sample consisted of around 4,000 HITs posted via the MTurk-Tracker during November and December 2015. The selected HITs were verified carefully and then added to the data set. In fact, we added only the HITs with complete metadata to ensure that there were enough keywords describing the HITs in order to prepare them for labeling and training.

After collecting the data, a Java-based desktop application was implemented to facilitate data labeling (see Fig. 3). One of the authors and two external researchers used the application to label the data set to the predefined types: Opinion Based (OB), Data Processing (DP), Content Generation (CG), Research Based (RB), and Content Conversion (CC). The final data set consists of 3,691 HITs. Fig. 4 illustrates the distributions of task types within the training sample.

To classify the tasks meaningfully, the extracted keywords, titles, and descriptions were subjected to a pre-processing. Some filters were applied in the following order:

1) Lowercase filter: Keywords are converted to lowercase letters to simplify the comparison.
2) Stemming filter: Stemming is the process of reducing words to their stems in order to minimize the number of distinct words. For example, stemming would reduce words like computer, computing, and compute to their stem, which is "comput". The iterated version of the Lovins Stemmer [53] was used for stemming in this work.
3) Stop-words removal filter: Stop words are words that are filtered out before the processing of textual data. An algorithm searches the text by a predefined list (stop-words) and deletes them from the text. Some of the common stop words are "the, is, at, but, be, been, and, as, out, ever, own, he, she, shall". A list of stop words based on Rainbow, a program that performs statistical text classification based on the Bow Library [54], was used in this work.



Task Topic: [illegible]
Task Description: [illegible]
Task Keywords: [illegible]

Agreement Cases: 87%
Disagreement Cases: 7%
Unsure Cases: 6%

FIGURE 3. Screenshots of the labeling application.

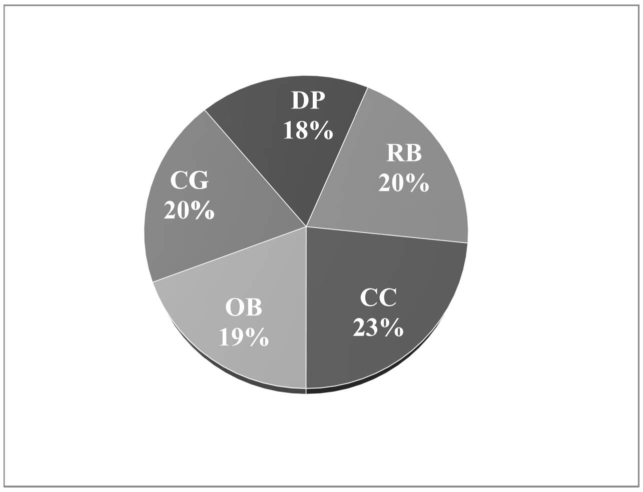


FIGURE 4. Training sample distribution.

2) RESULTS

After preprocessing, a machine-learning algorithm is used for learning how to identify task types. The goal is to build a component that is able to identify the task type accurately. The classification component was built on the task types provided earlier.

Machine-learning-based classification algorithms range in their theoretical background and application areas. The task of automatic text classification can be classified in three ways: unsupervised, supervised, and semi-supervised methods. Normally, supervised learning is used for automatic text classification [55]. Several algorithms were proposed for the supervised classifications of texts. Among these algorithms, Support Vector Machine (SVM), Naïve Bayes (NB), and k-Nearest Neighbor (k-NN) were shown to be the most appropriate in [55].

Hence, these three classifiers were tested: SVM, NB, and k-NN. The WEKA machine-learning toolkit [50] was used to test the classifiers using the features extracted from the data set. The component was evaluated using 10-fold cross validation over the labeled data.

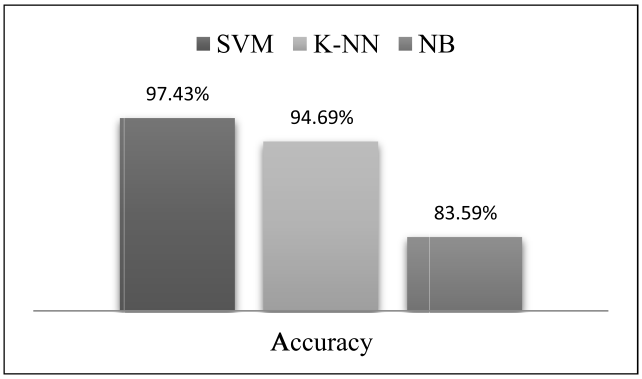


FIGURE 5. Accuracy of different classifiers.

As shown in Fig. 5, the SVM showed noticeable improvements over the k-NN and NB classifiers. The Precision and Recall of the three classifiers are displayed in Fig. 6.

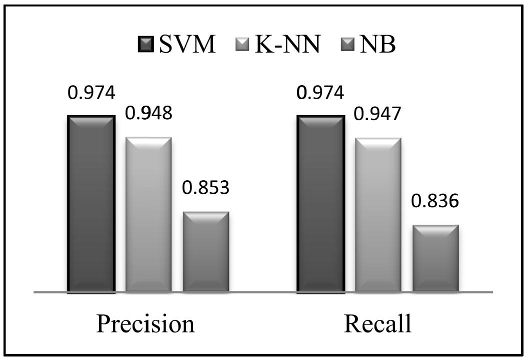


FIGURE 6. Precision and Recall of different classifiers.

Overall, the SVM is found to be effective in classifying the HITs to their types. It was able to classify 3,526 out of 3,619 HITs correctly (i.e., 97.43% accuracy). The classifier obtained a micro-average of precision of 0.974 and a recall of 0.974.

B. EXPERIMENT 2: SELECTING THE BEST-SUITED QCM FOR CROWDSOURCING

In this experiment, we evaluated the performance of the proposed approach and compared it against three quality control approaches: Random, StaticAMT, and ReputationBased.

The first approach simply selects one of the existing QCMs in a random way. The StaticAMT is a common approach that has been adopted by several crowdsourcing systems; however, it basically applies to only one QCM. The Amazon Turk crowdsourcing system, for example, has adopted this approach by using workers' voting mechanism to control and evaluate the execution of submitted tasks. The Reputation-Based approach collects users' ratings to build the reputation of each QCM and nominate the most reputable QCM to the target task regardless of its type. While this approach is more dynamic than the previous one, it was built under the assumption that one size fits all (i.e., in our context, this means that one QCM can handle and control all types of tasks).

Our proposed approach (the DynamicQCM) overcomes the limitations of the previous approaches by selecting the QCM dynamically. In other words, we consider the task features to select the best-suited QCM for a given task.

1) EXPERIMENTAL SETUP

The experiments were designed to simulate interactions between QCMs and tasks. The simulation allowed us to model different task profiles by varying task types, object types, and action types. It also allowed us to model different QCMs and generate requesters' ratings to reflect the performance of the used QCM with the executed task.

A population of different tasks and QCMs was created and added to the environment. Nine different tasks ($t_1 - -t_9$) were used in our experiments. Each task was identified by three features: <*task type, action type, object type*>. The values of these features were identified as follows:

(1) *Task type:* Task types of our data set include four of the identified and previously mentioned in section III-B-1: {OB (Opinion Based), DP (Data Processing), CG (Content Generation), and CC (Content Conversion)}. We named them accordingly as *{TT1, TT2, TT3, TT4}.*
(2) *Object type*: Inspired by the work in [52], the three most common object types used in our experiment are {Image, Text, and Audio} and are named {*O1, O2, and O3*}, respectively.
(3) *Action type:* Inspired by the top actions in the HITs collected from MTurk, we considered six types of action: {Classification, Writing, Describing, Translation, Summarization, and Transcription}. These actions are named {*A1, A2, A3, A4, A5, A6*}, respectively.

QCMs in our experiment were inspired from the literature. We used nine mechanisms for our study: Majority Voting, Peer Review, Iterative, Find-Fix-Verify, Tournament Selection Mechanism, I/O Agreement, Iterative Dual Pathway Structure, Worker Selection with Majority Voting, and Seeding with Iterative. They were named {*q1*, *q2*, *q3*, *q4*, *q5*, *q6*, *q7*, *q8*, *q9*}, respectively.

To test the performance of the approach, we generated 1,000 instances in the form of <*task type, object type, action type, QCM, rating*>. Each instance represents a task that was executed in the crowdsourcing system and controlled by a certain QCM. The generation of these instances was controlled by the values in Table 3. For example, <4, 2, 6, 9, 0.94> represents a rating of applying *q9*(Seeding with Iterative) with a task type *TT4*(Content Conversion), object type

TABLE 3. Tasks and best QCMs.

Task #	Task type	Object type	Action type	Best QCM
t_1	*TT1*	*O1*	*A1*	q_2
t_2	*TT2*	*O1*	*A1*	q_1
t_3	*TT2*	*O2*	*A1*	q_8
t_4	*TT3*	*O2*	*A2*	q_3
t_5	*TT4*	*O1*	*A1*	q_6
t_6	*TT4*	*O2*	*A4*	q_5
t_7	*TT4*	*O2*	*A5*	q_4
t_8	*TT4*	*O2*	*A6*	q_9
t_9	*TT4*	*O3*	*A6*	q_7

O2(Text), and action type *A6*(Transcription). The requester's rating generated for this instance was 0.94, meaning that the requester was happy with the result. In other words, the performance of *q9* was good enough to control the execution of this task and thereby produce satisfactory results for the requester. It may be noted that we allowed some variations in the generated values to reflect different task features that might be requested by the crowdsourcing users, and to reflect different QCMs that might be selected by the system. Furthermore, we allowed some variations in ratings to mirror the real-world situations where the requesters may have similar tasks and QCMs; however, they may provide different ratings.

Overall, the data set was diversified across various task types, objects, actions, and QCMs. The generated ratings were controlled to give a high level of rating if the best-suited QCM was used with the target task, as in Table 3, while giving an average or low rating to other QCMs. For example, the simulator would generate a level of rating of approximately 0.90 whenq_1 is used to control $t_{2,}$ while a level of rating of approximately 0.3 would be generated when the same QCM is used with t_9. The performance of QCMs over various crowdsourcing tasks is illustrated in Fig. 7.

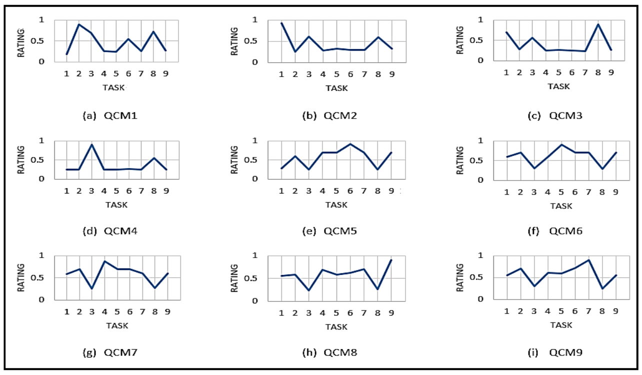


FIGURE 7. Performance of QCMs over various crowdsourcing tasks.

2) RESULTS

Selecting the best QCM for t$_2$. In this experiment, we examined how data size (ratings) might affect the selection of the best QCM for task (t_2) using the four approaches. For this purpose, we repeatedly selected the QCM by each approach every additional 100 instances (i.e., at 100, 200, 300, *etc.*). The results of this experiment are shown in Fig. 8.

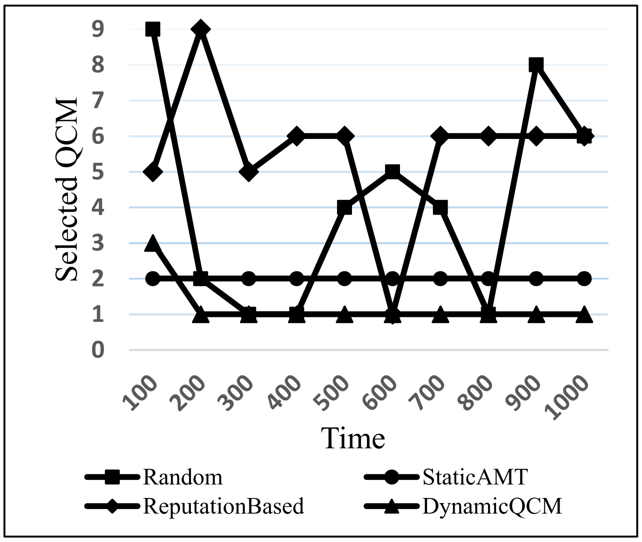


FIGURE 8. Selected QCM for t2 over time.

As can be seen from Fig. 8, the Random approach did not fit with a particular QCM in the test. Its selection was not affected by the data size at all. This is because this approach selects the QCM in a random way regardless of any other factors, such as task type. The random approach stuck by chance with the best QCM (q_1) three times at data points 300, 400, and 800.

The StaticAMT approach was always stuck with selecting q_2 for this task regardless of the population of the data. That is because StaticAMT does not consider task types and uses only one QCM, Majority Voting, to evaluate all types of tasks. For this reason, StaticAMT would keep selecting q_2 as the best choice for t_2 and would never select q_1.

The ReputationBased approach did not converge to a particular QCM in the test. This is because its selection was affected by the criterion to identify the most reputable QCM at each data point. The ReputationBased approach selected q_1 as the best QCM for t_2 only once, at data point 600. This is because at this time q_1 was the most reputable QCM in the system. Although this approach takes into account the user feedback to determine the best QCM, it does not consider the task type, making its selection generic and irrational.

The DynamicQCM approach initially picked$q_3$as the best QCM for t_2. This is because the system still did not have enough data about the performance of q_1 and t_2. The selection was heavily affected by the available data. After 200 tasks, however, the DynamicQCM approach was able to stick with the best-suited mechanism (q_1) and kept selecting it afterwards.

Fig. 9 illustrates the accuracy of QCM selection for the four approaches. It is clear that our proposed approach achieved higher accuracy selection compared to the three other approaches. It may be noted that this experiment was highly sensitive to the selected task. If we repeat this experiment with t_1, for example, the StaticAMT approach would achieve 100% accuracy selection. This is because the Stati-

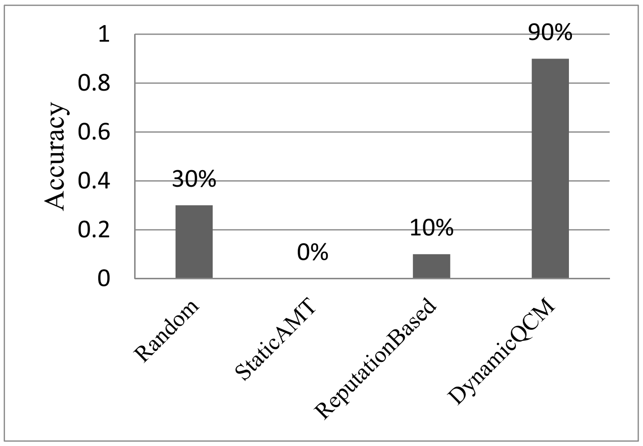


FIGURE 9. Accuracy of QCM selection for the four approaches.

cAMT approach applies q_2 every time, regardless of the type of task. However, as illustrated in Table 3, q_2 is the best-suited mechanism to control and evaluate t_1.

Selecting the best QCM for each task at time 1,000. In this experiment, we examined the selection success rate for each approach at time 1,000. That is, we measured the rate of selecting the "best" QCM for each task by the four approaches. We used the same data sets of the previous experiment and calculated the successful selection rate according to Eq. (1), where $\emptyset_c$ is the number of correct selections of QCM, and $\emptyset$ is the total number of user requests to the crowdsourcing system (nine in our experiment).

$$Selectionsuccessrate = \frac{\emptyset_c}{\emptyset}. \quad (1)$$

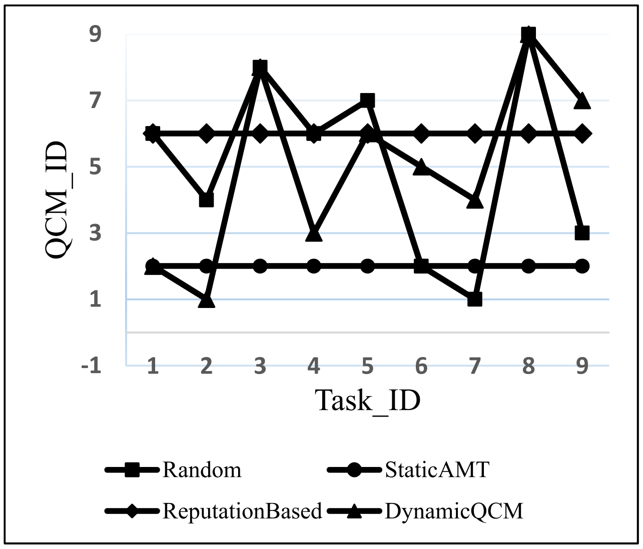


FIGURE 10. Selected QCM for each task by the four approaches at time 1,000.

Fig. 10 shows the selected QCM by the four approaches for each task at time 1,000.

As can be seen from Fig. 10, the selection of the Random approach was basically random. Its selection of the best-suited QCM was correct on two instances by chance: for $t_3$and t_8. Overall, as it is a blind approach, the selection success rate of Random was very low (20%, as shown in Fig. 11).

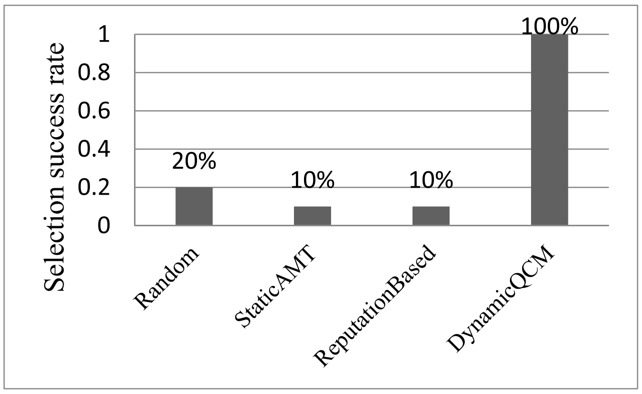


FIGURE 11. Selection success rate for the four approaches at time 1,000.

The StaticAMT approach selected $q_2$for all nine tasks for the same reason given in the previous experiment. It was correct only when the task was t_1, as its best QCM was q_2. As shown in Fig. 11, the selection success rate for this approach was very low (only 10%). This is because the StaticAMT approach did not consider the task type in its selection and kept selecting the same QCM regardless of the type of requested task.

The ReputationBased approach selected q_6 for all nine tasks because the q_6 had the highest reputation score over all of the other QCMs at time 1,000. It is evident that the ReputationBased approach was not able to stick with the correct QCM in almost all tasks. The approach selected the best QCM accurately with t_5 because q_6 was the best-suited QCM for this kind of task. Similar to the previous approach, its selection success rate was very low.

The DynamicQCM approach was able to select the best-suited QCM for all types of tasks. By utilizing the Task_QCM ontology that was enriched by the QCM reputation engine, this approach was able to adapt its selection based on the type of task. In contrast to the previous approaches, the selection of the DynamicQCM approach was very accurate, and it achieved a 100% success rate. It may be noted that this approach is sensitive to the data set collected from the crowdsourcing requesters, and the reputation of the various QCMs can be affected by the size of the collected data. As we saw in the previous experiment, 100 instances were not enough to discover and nominate the best-suited QCM for t_2. However, after collecting more data (ratings from the requester), the DynamicQCM was able to stick with the correct one.

V. DISCUSSION

There are many issues that need to be considered in the proposed approach. First, it is assumed that a new QCM can be added at any given point in time. The challenge is to match this new QCM to the tasks in the absence of its performance records. This problem is called the "cold start" in reputation systems. Solutions that were proposed in [56] and [57] can be adapted and reused to alleviate the cold start in crowdsourcing systems.

Second, the proposed approach uses historical information to calculate the reputation score of quality control mechanisms. However, it is known that the reliability of users' ratings is a challenge in reputation systems [58]. Many mechanisms have been proposed in the literature to detect and remove unfair ratings [59]–[61]. In the context of crowdsourcing, in contrast to a highly competitive e-commerce system, there is no motivation or direct value for users to provide biased or unfair ratings to deviate the reputation score of a given QCM.

Third, the reliability of the produced reputation score is essential for providing a rational mapping between a given task and the best-suited QCM. In other words, if the historical information collected for a given QCM is inadequate, the selected mechanism may work ineffectively. In the QCM reputation engine, each data item used provides an independent evidence on the reliability of the produced reputation score; therefore, intuitively, the more the evidence we have, the higher the confidence we would have for the reputation score [62]. Thus, it is important to look for an alternative source of data when there are few ratings available for the QCM reputation engine. This problem is called a "sparsity" problem in other domains [63], [64]. In our context, the mapper can utilize the Task_QCM ontology in order to use the data that are relevant to similar tasks to produce a more reliable selection. Inspired by the mechanisms used in personalized reputation systems [61], the selection of the QCM can be modeled by combining the weighted reputation scores of the QCMs for both the current and similar tasks.

Fourth, for each new request to the crowdsourcing system, the proposed approach seeks out a similar task and then nominates the QCM associated with it to evaluate and control the new one. However, in some cases, the most similar task is associated with a QCM that has a low reputation score (e.g., t_8 in Table 2). Our approach in this case is to select a QCM randomly (blind selection). Instead, to enable a rational selection, it might be better to check the next similar task and nominate the associated QCM if its reputation score is high.

Finally, there is a need to consider how to identify task types of new tasks when key metadata are missing. In such cases, instead of using a task classifier to classify the new task, an intervention of an ontology expert might be required. Some important numerical values, such as a threshold of the classifier to identify a new domain, or just a similar instance, should be introduced in the Task_QCM ontology.

VI. CONCLUSION AND FUTURE WORK

In this paper, we proposed a dynamic approach for selecting the best-suited QCM for a task rather than selecting a specific QCM for all types of tasks. The approach relies on three components: (1) a task classifier that identifies the types of crowdsourcing tasks automatically based on their features; (2) a Task_QCM ontology implemented to standardize the domain, which allows the sharing and reusing of knowledge and supports the identification of semantic similarities of tasks; and (3) a QCM reputation engine, which is responsible for the calculation of QCM reputations.

Our evaluation consists of two main parts. First, we presented an evaluation of the task classifier based on the real data collected from MTurk. Second, we conducted a set of experiments to compare our DynamicQCM approach against other quality control approaches using synthetic data generated from a simulation environment. Overall, the results show that the SVM classifier is quite effective in classifying the tasks to their types and obtained an accuracy of 97.43%. Experiments carried out to compare our approach against three other approaches demonstrated that our approach can provide better results.

As future steps, we would like to extend the ontology to store new features of the task and study their relations with QCMs. To illustrate, we would like to explore the impact of a reward mechanism on the selection of the QCM, or the impact of the domain of the task (e.g., education, sports, *etc.*) on the classification of task types. Furthermore, we intend to use the data generated from our simulator as training data, where the task features (task type, object type, and action type) with their QCM could be learned and used offline.

In sum, the performance of our proposed approach becomes better and more precise depending on the number of users. The more the users we have, the greater the increase of the knowledge base is, and, consequently, the broader the system becomes.

REFERENCES

[1] L. Ponciano, F. Brasileiro, N. Andrade, and L. Sampaio, "Considering human aspects on strategies for designing and managing distributed human computation," *J. Internet Services Appl.*, vol. 5, no. 1, p. 10, 2014.

[2] N. Luz, N. Silva, and P. Novais, "Generating human-computer micro-task workflows from domain ontologies," in *Human-Computer Interaction. Theories, Methods, and Tools* (Lecture Notes in Computer Science), vol. 8510. Cham, Switzerland: Springer, 2014, pp. 98–109.

[3] A. Kittur *et al.*, "The future of crowd work," in *Proc. Conf. Comput. Supported Cooperative Work (CSCW)*, 2013, pp. 1301–1317.

[4] M. Allahbakhsh, A. Ignjatovic, B. Benatallah, S. M. R. Beheshti, E. Bertino, and N. Foo, "Reputation management in crowdsourcing systems," in *Proc. 8th IEEE Int. Conf. Collaborative Comput., Netw., Appl. Workshop*, Oct. 2012, pp. 664–671.

[5] D. E. Difallah, G. Demartini, and P. Cudré-mauroux, "Pick-a-crowd: Tell me what you like, and i'll tell you what to do," in *Proc. Int. World Wide Web Conf. Committee (IW3C2)*, 2013.

[6] A. Kittur, E. H. Chi, and B. Suh, "Crowdsourcing user studies with mechanical turk," *Proc. 26th Annu. CHI Conf. Hum. Factors Comput. Syst. (CHI)*, 2008, pp. 453–456.

[7] N. Luz, N. Silva, and P. Novais, "A survey of task-oriented crowdsourcing," *Artif. Intell. Rev.*, vol. 44, no. 2, pp. 187–213, Aug. 2015.

[8] 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 Comput.*, vol. 17, no. 2, pp. 76–81, Mar. 2013.

[9] H. J. Khasraghi and A. Aghaie, "Crowdsourcing contests: Understanding the effect of competitors' participation history on their performance," *Behav. Inf. Technol.*, vol. 33, no. 12, pp. 1383–1395, Mar. 2014.

[10] Y. Zhao and Q. Zhu, "Evaluation on crowdsourcing research: Current status and future direction," *Inf. Syst. Frontiers*, vol. 16, no. 3, pp. 417–434, Apr. 2012.

[11] MTurk. (2016). *MTurk: Amazon Mechanical Turk*. Accessed: Jan. 1, 2016. [Online]. Available: http://www.mturk.com/

[12] W. Safire. (2009). On language. New York Times Mag. Accessed: Mar. 24, 2016. [Online]. Available: http://www.nytimes.com/2009/02/08/magazine/08wwln-safire-t.html?_r=3&ref=magazine&

[13] J. Howe. (Jun. 2006). The Rise of Crowdsourcing. WIRED Mag. Accessed: Sep. 12, 2015. [Online]. Available: https://www.wired.com/2006/06/crowds/

[14] A. Doan, R. Ramakrishnan, and A. Y. Halevy, "Crowdsourcing systems on the World-Wide Web," *Commun. ACM*, vol. 54, no. 4, pp. 86–96, Apr. 2011.

[15] J. Howe, *Crowdsourcing: Why the Power of the Crowd is Driving the Future of Business*. New York, NY, USA: Crown Business, 2008.

[16] D. C. Brabham, "Crowdsourcing as a model for problem solving: An introduction and cases," *Int. J. Res. New Media Technol*, vol. 14, no. 1, pp. 75–90, Feb. 2008.

[17] E. Estellés-Arolas, F. González-Ladrón-De-Guevara, "Towards an integrated crowdsourcing definition," *J. Inf. Sci.*, vol. 38, no. 2, pp. 189–200, Apr. 2012.

[18] D. Schall, "Automatic quality management in crowdsourcing [leading edge]," *IEEE Technol. Soc. Mag.*, vol. 32, no. 4, pp. 9–13, Dec. 2013.

[19] J. Wang, P. G. Ipeirotis, and F. Provost, "A framework for quality assurance in crowdsourcing," NYU, New York, NY, USA, Working Paper 2451/31833, 2013.

[20] (2016). *Upwork*. Accessed: Mar. 24, 2016. [Online]. Available: https://www.upwork.com/

[21] (2016). *Freelancer*. Accessed: Mar. 24, 2016. [Online]. Available: https://www.freelancer.com/

[22] N. Savage, "Gaining wisdom from crowds," *Commun. ACM*, vol. 55, no. 3, pp. 13–15, Mar. 2012.

[23] G. Markowsky, "Crowdsourcing, big data and homeland security," in *Proc. IEEE Int. Conf. Technol Homeland Sec. (HST)*, Nov. 2013, pp. 772–778.

[24] M. Sabou, K. Bontcheva, and A. Scharl, "Crowdsourcing research opportunities: Lessons from natural language processing," in *Proc. 12th Int. Conf. Knowl. Manage. Knowl. Technol.*, 2012, p. 8.

[25] (2016). *CastingWords*. Accessed: Mar. 24, 2016. [Online]. Available: https://castingwords.com/

[26] M. Franklin, D. Kossman, T. Kraska, S. Ramesh, and R. Xin, "CrowdDB: Answering queries with crowdsourcing," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2011, pp. 61–72.

[27] G. Kazai, "An Exploration of the Influence that Task Parameters have on the Performance of Crowds," in *Proc. CrowdConf.*, 2010.

[28] (2016). *Crowdflower*. Accessed: Jan. 4, 2016. [Online]. Available: http://www.crowdflower.com/

[29] P. G. Ipeirotis, F. Provost, and J. Wang, "Quality management on Amazon mechanical turk," in *Proc. ACM SIGKDD Workshop Hum. Comput. (HCOMP)*, 2010, p. 64.

[30] K. El Maarry, W. Balke, H. Cho, S. Hwang, and Y. Baba, "Skill ontology-based model for quality assurance in crowdsourcing," in *Database Systems for Advanced Applications* (Lecture Notes in Computer Science), Berlin, Germany: Springer, vol. 8505. 2014, pp. 376–387.

[31] G. Kazai, J. Kamps, and N. Milic-Frayling, "An analysis of human factors and label accuracy in crowdsourcing relevance judgments," *Inf. Retr.*, vol. 16, no. 2, pp. 138–178, Jul. 2013.

[32] A. Quinn and B. Bederson, "Human computation: A survey and taxonomy of a growing field," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst. (CHI)*, 2011, pp. 1403–1412.

[33] D. Iren and S. Bilgen, "Cost models of quality assurance in crowdsourcing," in *Proc. IEEE 5th Int. Conf. Commun. Electron. (ICCE)*, Jul. 2014, pp. 504–509.

[34] A. Sorokin and D. Forsyth, "Utility data annotation with Amazon mechanical turk," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshop (CVPRW)*, Jun. 2008, pp. 1–8.

[35] Y.-A. Sun, S. Roy, and G. Little, "Beyond independent agreement: A tournament selection approach for quality assurance of human computation tasks," in *Proc. Hum. Comput.*, 2011, pp. 113–118.

[36] M.-C. Yuen, I. King, and K.-S. Leung, "A survey of crowdsourcing systems," in *Proc. IEEE 3rd Int. Conf. Privacy, Secur., Risk Trust IEEE Proc. 3rd Conf. Soc. Comput.*, Oct. 2011, pp. 766–773.

[37] G. Little, L. B. Chilton, M. Goldman, and R. C. Miller, "Exploring iterative and parallel human computation processes," in *Proc. ACM SIGKDD Workshop Hum. Comput. (HCOMP)*, 2010, p. 68.

[38] T. Hobfeld *et al.*, "Survey of Web-based crowdsourcing frameworks for subjective quality assessment," in *Proc. 16th Int. Workshop Multimedia Signal Process. (MMSP)*, Sep. 2014, pp. 22–24.

[39] M. S. Bernstein *et al.*, "Soylent: A word processor with a crowd inside," in *Proc 23rd Annu. ACM Symp. User Interface Softw. Technol.*, 2010, pp. 313–322.

[40] B. Liem, H. Zhang, and Y. Chen, "An iterative dual pathway structure for speech-to-text transcription," in *Proc. AAAI Hum. Comput. Workshop (HCOMP)*, San Francisco, CA, USA, 2011, pp. 1–6.

[41] L. Hetmank, "A lightweight ontology for enterprise crowdsourcing," in *Proc. 22nd Eur. Conf. Inf. Syst. (ECIS)*, 2014, p. 886.

[42] D. E. Difallah, M. Catasta, G. Demartini, P. G. Ipeirotis, and P. Cudré-Mauroux, "The dynamics of micro-task crowdsourcing: The case of Amazon mturk," in *Proc. 24th Int. Conf. World Wide Web*, 2015, pp. 238–247.

[43] U. Gadiraju, R. Kawase, and S. Dietze, "A taxonomy of microtasks on the Web," in *Proc. 25th ACM Conf. Hypertext Soc. Media*, 2014, pp. 218–223.

[44] R. Alabdujabbar and H. Al-Dossari, "Towards a classification model for tasks in crowdsourcing," in *Proc. 2nd ACM Int. Conf. Internet Things, Data Cloud Comput. (ICC)*, Cambridge, U.K., 2017, p. 23.

[45] R. Alabduljabbar and H. Al-Dossari, "Ontology for task and quality management in crowdsourcing," *Int. J. Comput.*, vol. 22, no. 1, pp. 90–102, 2016.

[46] T. Slimani, "Description and evaluation of semantic similarity measures approaches," *Int. J. Comput. Appl.*, vol. 80, no. 10, pp. 25–33, 2013.

[47] M. Levandowsky and D. Winter, "Distance between sets," *Nature*, vol. 234, no. 5323, pp. 34–35, 1971.

[48] S. Schnitzer, S. Neitzel, S. Schmidt, and C. Rensing, "Perceived task similarities for task recommendation in crowdsourcing systems," in *Proc. 25th Int. Conf. Companion World Wide Web*, 2016, pp. 585–590.

[49] A. Singhal, "Modern information retrieval: A brief overview," *IEEE Comput. Soc. Tech. Committee Data Eng.*, vol. 24, no. 4, pp. 35–43, Jan. 2001.

[50] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten, "The WEKA data mining software: An update," *ACM SIGKDD Explor. Newslett.*, vol. 11, no. 1, pp. 10–18, 2009.

[51] *Amazon Mechanical Turk Requester UI Guide (API Version 2014-08-15)*. Accessed: Jan. 1, 2016. [Online]. Available: http://docs.aws.amazon.com/AWSMechTurk/latest/RequesterUI/OverviewofMturk.html

[52] P. G. Ipeirotis, "Analyzing the Amazon mechanical turk marketplace," *ACM XRDS*, vol. 17, no. 2, p. 16, Dec. 2010.

[53] (2016). *The (Un)Official Lovins Stemmer Page*. Accessed: Jan. 1, 2016. [Online]. Available: http://www.cs.waikato.ac.nz/~eibe/stemmers/index.html

[54] A. K. McCallum. (1996). *Bow: A Toolkit for Statistical Language Modeling, Text Retrieval, Classification and Clustering*. Accessed: Jan. 1, 2016. [Online]. Available: http://www.cs.cmu.edu/~mccallum/bow

[55] A. Khan, B. Baharudin, L. H. Lee, and K. Khan, "A review of machine learning algorithms for text-documents classification," *J. Adv. Inf. Technol.*, vol. 1, no. 1, pp. 4–20, 2010.

[56] X. N. Lam, T. Vu, T. D. Le, and A. D. Duong, "Addressing cold-start problem in recommendation systems," *Proc. 2nd Int. Conf. Ubiquitous Inf. Manage. Commun.*, 2008, pp. 208–211.

[57] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock, "Methods and metrics for cold-start recommendations," in *Proc. 25th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr. (SIGIR)*, vol. 46, 2002, pp. 253–260.

[58] A. Jøsang, R. Ismail, and C. Boyd, "A survey of trust and reputation systems for online service provision," *Decis. Support Syst.*, vol. 43, no. 2, pp. 618–644, 2007.

[59] L. Zhang, S. Jiang, J. Zhang, and W. K. Ng, "Robustness of trust models and combinations for handling unfair ratings," in *Proc. Int. Conf. Trust Manage. (IFIP)*. Berlin, Germany: Springer, 2012, pp. 36–51.

[60] S. Liu, H. Yu, C. Miao, and A. C. Kot, "A fuzzy logic based reputation model against unfair ratings," in *Proc. Int. Conf. Auton. Agents Multi-Agent Syst.*, 2013, pp. 821–828.

[61] J. Zhang and R. Cohen, "A personalized approach to address unfair ratings in multiagent reputation systems," in *Proc. AAMAS Work. Trust Agent Soc.*, 2006, pp. 1–10.

[62] H. Al-Dossari and J. Shao, "Modelling confidence for quality of service assessment in cloud computing," in *Proc CONF-IRM*, 2013, p. 48.

[63] Z. Huang, H. Chen, and D. Zeng, "Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering," *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 116–142, 2004.

[64] M. Papagelis, D. Plexousakis, and T. Kutsuras, "Alleviating the sparsity problem of collaborative filtering using trust inferences," in *Proc. 3rd Int. Conf. Trust Manage.*, 2005, pp. 224–239.

REHAM ALABDULJABBAR received the B.Sc. degree in computer applications from King Saud University, Saudi Arabia, in 2005, the M.Sc. degree in information technology from The University of Adelaide, Australia, in 2007, and the Ph.D. degree in information systems from King Saud University, in 2017, where she is currently an Assistant Professor with the Information Technology Department, College of Computer and Information Sciences. She has many publications in international journals and conferences. Her research interests include human computation, crowdsourcing, and big data analytics.

HMOOD AL-DOSSARI received the B.S. and M.S. degrees in computer science from King Saud University, Saudi Arabia, in 1998 and 2005, respectively, and the Ph.D. degree in computer science from Cardiff University, in 2011. He is currently an Associate Professor with the Information Systems Department, College of Computer and Information Sciences, King Saud University. He has many publications in international journals and conferences. His research interests include quality of service assessment, social mining, human–computer interaction, and reputation and trust systems. He has attended various conferences and presented many seminars.