Erscheint in: International Journal of Cooperative Information Systems (IJCIS) 2012 ISSN 0218-8430

DYNAMIC AND GOAL-BASED QUALITY MANAGEMENT FOR HUMAN-BASED ELECTRONIC SERVICES

Robert Kern
Karlsruhe Institute of Technology (KIT)

Hans Thies
SAP Research, CH 9000 St. Gallen

Christian Zirpins
Karlsruhe Institute of Technology (KIT)

Gerhard Satzger
Karlsruhe Institute of Technology (KIT)

March 5, 2012

Abstract

Crowdsourcing in the form of human-based electronic services (people services) provides a powerful way of outsourcing tasks to a large crowd of remote workers over the Internet. Research has shown that multiple redundant results delivered by different workers can be aggregated in order to achieve a reliable result. However, basic implementations of this approach are rather inefficient as they multiply the effort for task execution and are not able to guarantee a certain quality level. In this paper we are addressing these challenges by elaborating on a statistical approach for quality management of people services which we had previously proposed. The approach combines elements of statistical quality management with dynamic group decisions. We present a comprehensive statistical model that enhances our original work and makes it more transparent. We also provide an extendible toolkit that implements our model and facilitates its application to real-time experiments as well as to simulations. A quantitative analysis based on an optical character recognition (OCR) scenario confirms the efficiency and reach of our model.

Keywords: Crowdsourcing, statistical quality control, human computation, eServices, people services

1 Introduction

The general idea of human-based electronic services is to leverage Web service technology in order to access and tap the crowd of Internet users for human workforce. The continuous popularity of Amazon's Mechanical Turk (MTurk) platform[1] and the large number of companies that are basing their business model entirely on that platform demonstrate the potential of this approach. The MTurk platform acts as a broker between requesters who publish human intelligence tasks (HITs) and workers who solve tasks in return for a typically small monetary compensation. The most popular types of tasks on MTurk are transcribing recorded speech, generating content, classifying or categorizing items (e.g. images), collecting data and providing feedback (e.g. regarding Web sites) [1].

Kern et al. proposed the term *people services (pServices)* for this type of human-based electronic services [2]. As pServices enable the allocation of human resources on demand, they promise to enhance the workforce scalability and lower fixed labor costs by reducing the cost overhead (salaries, workplace costs etc.) in times of low demand. However, they also pose problems of limited control over individual contributors which might compromise the quality of work results. This raises questions about adequate quality management mechanisms for pService scenarios and their application in the context of today's pService platforms.

A general approach to quality assurance that is heavily used in practice and can be applied to a broad set of pServices scenarios is the *majority vote* (MV) mechanism, which introduces redundancy by passing the same task to multiple workers and aggregating the results in order to compute the result with the highest probability for correctness [3]. Existing MV applications typically apply a fixed level of redundancy to each individual task, i.e. each task is performed by multiple workers. From the perspective of quality management that means that the quality of each individual task is validated. However, concepts of *statistical quality control (SQC)* show that the quality management effort can usually be drastically reduced by taking only samples rather than by performing a full inspection of all individual items [4]. Moreover, a fixed degree of redundancy is both inefficient and incapable of assuring a certain level of result quality because the level of agreement (and so the expected result quality) varies depending on the failure rates of the involved workers. Exploiting these potentials, we propose a quality management (QM) approach for pServices, which improves the traditional MV approach in three ways:

- It reduces the QM effort in a *horizontal* direction by validating only a sample of tasks rather than all tasks.
- It reduces the QM effort in a *vertical* direction by dynamically adjusting the level of redundancy rather than working with a fixed level of redundancy.

[1] `http://www.mturk.com/`

- It allows to guarantee a certain quality level.

Within the multifaceted dimensions of quality, our approach concentrates on the correctness dimension as the ability to return a minimum percentage of results that are free of errors [2]. According to Juran's definition of quality as *fitness for use* [5], the paper assumes that the service requester can clearly categorize a task result as correct or incorrect. The level of correctness is determined by a comparison with the ideal result (gold standard) provided by the service requester.

While our QM approach for pService was first introduced at ICSOC 2010 [6], this paper extends the underlying statistical model and puts it on a solid foundation. Furthermore, a software toolkit was developed which is introduced by this paper. The toolkit has enabled us to conduct live experiments on MTurk in addition to the simulation runs that we had studied before.

After introducing the SQC fundamentals in section 2, section 3 presents the QM approach for pServices. Section 4 describes the implementation of a QM software toolkit for pService that has been used to evaluate the QM approach based on an optical character recognition (OCR) scenario on top of the MTurk platform. The results are provided in section 5. The paper closes with related work, a discussion section and a summary and outlook in sections 6 through 8.

2 Fundamentals

This chapter describes some fundamentals of SQC required for developing the QM approach in section 3.

2.1 Acceptance sampling

Acceptance Sampling is the process to decide based on a sample whether a set of units meets certain quality requirements or not. Acceptance sampling determines the probability of a lot of units being within the specified quality levels, and accepts or rejects lots based on its quality characteristics. A sampling plan is a procedure where a sample of n units is drawn from a lot of size N. If the number of defects in the sample is higher than the *acceptance number* c, the lot is rejected. Otherwise it is accepted. If the units do not occur in batches, but in a continuous production, such as in line assembly or in a service scenario, the process has to be decomposed into artificial batches. However, before a whole batch has been handled, quality levels for this batch cannot be guaranteed and the results of this batch cannot be further processed. In order to overcome this restriction, *continuous sampling plans* have been developed.

2.2 Continuous Sampling Plans

Continuous Sampling Plans (CSPs) control the inspection frequency and replacement of defects in such a way that the outgoing stream of items exceeds a certain average level of quality. Dodge developed the first continuous sampling

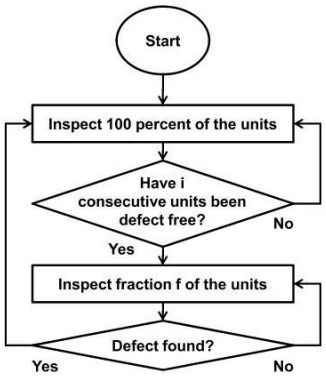

Figure 1: Procedure of the continuous sampling plan CSP-1.

plan, the CSP-1. This plan has been further developed and adapted by Dodge et. al and Lieberman et al. amongst others [7, 8]. The probably most cited and most used continuous sampling plan still is the CSP-1. The reason is not only its relative simplicity, but also its efficiency, which in few cases is exceeded by other continuous sampling plans like the CSP-2 [9]. Dodge made the following assumptions developing the CSP-1:

- Process of incoming units is under statistical control and follows a Bernoulli distribution.
- Sample inspection is perfect.
- Defective units are replaced by good ones.

The sampling plan is designed for attributes, thus quality parameters are categorized as either good or defective. This means that if the incoming process is under statistical control i.e. the incoming fraction defective p does not change over time, the process can be described by a Bernoulli process with defect probability p. As illustrated by figure 1, the sampling plan starts with a 100% inspection phase. Once i consecutive units have been found free of defects, the CSP-1 switches to a fractional inspection mode in which only a fraction f of the units are inspected. As soon as a defective unit is found, the model returns to 100% inspection, and so on. Defective units are either reworked or replaced with good ones. The parameter i is called the *clearance number* and f is called the *sampling fraction* [4]. The most important characteristic of the CSP-1 is the *average outgoing quality limit* ($AOQL$) [4]. It is the highest average amount of defective units that might be passing through without being inspected that can be reached depending on the incoming fraction defective p.

There are multiple combinations of the parameters i and f which result in the same value of $AOQL$. For a given $AOQL$, one of the parameters i and f can be chosen depending on the needs of the scenario and the corresponding

parameter can then be calculated according to (2.1):

$$f = \left(1 + \frac{\left(1 + \frac{1}{i}\right)^{i+1} \cdot i \cdot AOQL}{(1 - AOQL)^{i+1}}\right)^{-1} \tag{2.1}$$

The following chapters describe two ways of choosing i and f which will be needed in the remainder of the paper.

2.2.1 Short sequences of low quality

One possible objective of choosing i is to make the CSP-1 resistant against short sequences of low quality [10]. Let P^* be the quality level of a series of l items within the continuous stream of items. In order to ensure that the probability of accepting such a series is not higher than $\tilde{w}$, the sampling fraction f must be

$$f = \frac{1}{P^*}\left(1 - \tilde{w}^{\frac{1}{l}}\right) . \tag{2.2}$$

The corresponding clearance number i can be identified with (2.1) or (2.3) by inserting different values of i until an f close to the desired one has been found. As i must be a whole-number, only discrete values of f are possible.

2.2.2 Imperfect sample inspection

In many scenarios, the inspection of the samples is not perfect but an *inspection error* applies to the sampling process. In case of imperfect inspection, two types of inspection errors can be made:

- E_1: A good item is classified as defective (type 1 inspection error).
- E_2: A defective item is classified as good (type 2 inspection error).

Under the assumption of imperfect inspection, (2.1) is replaced by (2.3): [11]

$$f = \frac{(1 - (1 - e_2)\hat{p} - (1 - \hat{p})e_1)^i \cdot (1 - \frac{AOQL}{\hat{p}})}{((1 - (1 - e_2)\hat{p} - (1 - \hat{p})e_1)^i - 1) \cdot (1 - \frac{AOQL}{\hat{p}}) + (1 - e_2)} \tag{2.3}$$

The parameters $e_1 = P(E_1)$ and $e_2 = P(E_2)$ are the probabilities of a type 1 and type 2 inspection error. $AOQL$ is the average outgoing quality limit and $\hat{p}$ is the expected incoming fraction defective, i.e. the probability of processing a defective item.

3 Statistical Quality Control for pServices

This chapter describes our statistical quality management approach for pServices. After introducing the underlying assumptions in section 3.1 and providing an overview in section 3.2, the detailed process flow is covered by section 3.3. Section 3.4 explains how statistical quality control is used within the model and section 3.5 introduces the extended *dynamic majority vote* (DMV) approach which is the main contribution of the paper.

3.1 Assumptions

The basic scenario comprises three roles: The *service requester*, the *crowdsourcing platform* and the *workers*. The platform acts as an intermediary between the requester who publishes tasks and workers who pick tasks and work on them in return for a compensation per task. There is typically a large number of equivalent tasks of the same *task type* that consist of the same *task description* but different *task data*. The task description primarily contains the instructions for the workers on how to perform the task as well as information about the expected result quality. The task data is the variable part which might represent different pictures to be annotated, different addresses to be validated or different products to be classified. The paper makes some additional assumptions about the underlying platform, the tasks as well as the workers.

3.1.1 Platform assumptions

In accordance with existing crowdsourcing platforms like MTurk it is assumed that the platform allows for tracking individual workers based on an individual *worker ID* which is returned to the requester for each response delivered by the worker. The platform also supports multiple redundant *assignments* of the same task and ensures that they are completed by different workers in order to use the responses for group decisions. There are also means for defining worker pools i.e. for making specific tasks only available to a subset of the workers. MTurk implements that by offering so-called *qualification tests* which also ensure that the workers having a certain qualification fall at least initially below a certain failure rate when working on a specific type of tasks. The lower two boxes of figure 2 give an overview of the basic crowdsourcing process.

3.1.2 Task assumptions

As the quality management approach presented here is a form of a majority vote (MV) approach, it is restricted to *deterministic tasks* i.e. to such tasks for which a certain well-defined optimal result is defined [12]. Therefore, redundant responses delivered by multiple workers for assignments of the same task can be compared to each other or can be aggregated into a single consolidated result. It is further assumed that a large number of similar tasks is available that require similar skills and that a fixed compensation is paid to the workers for each task independent of the quality of the work results.

3.1.3 Worker assumptions

From a statistical perspective it is important that the workers are working independently of each other. Most important, a worker who is working on a specific task must not collaborate (e.g. agree on a response) with other workers working on assignments of the same task. Therefore, there should be a large crowd of workers who do not know each other. Another assumption is that a failure rate can be attached to each worker which is the same for all tasks of a given task

type and which only slowly changes over time, for example because workers are improving their skills or because they are getting tired after performing a large number of tasks.

3.2 Model overview

The objective of the model described in this paper is to leverage statistical quality control in order to guarantee a certain long-run average outgoing quality for a continuous stream of crowdsourcing results, while the inspection costs in terms of labor work are minimized. As we assume a fixed payment per task, the QM costs can be minimized by minimizing the total number of assignments. The model can be seen as a quality management component on top of the basic crowdsourcing platform outlined in section 3.1. The model consists of two functional parts: A statistical quality control component and a sample inspection component.

The statistical quality control component uses the continuous sampling plan (CSP-1) with the extension of imperfect inspection presented in section 2. Because the workers are acting independently from each other, the sampling process has to be performed at worker-level. The same $AOQL$ is applied to all workers who work on the same type of task i.e. the same response quality is requested from all participating workers.

The CSP-1 requires a mechanism for sample inspection. For this purpose, a *dynamic majority vote* approach (DMV) was designed which will be described in detail in the following section. The DMV dynamically increases the redundancy by including additional workers in the MV decision until a predefined inspection quality level φ_{min} is reached.

Although the difficulty of all tasks is assumed to be similar, there will always be outliers which are harder to solve or which are even not solvable at all. Those tasks are escalated back to the requester if they fall below a predefined *escalation limit* ε_{min}. That way, the requester can use the information to improve the task design.

3.3 Process flow

The overall scenario is given by figure 2: A requester submits a task to the quality management component which immediately publishes it to the crowdsourcing platform. A worker grabs the task, works on it and returns a response which is fed back to the QM component in combination with the worker's ID. Depending on the CSP-1 status of that worker a sample inspection is initiated or not. If not, the QM component accepts the worker's response as the final result and passes it back to the requester without further validation. In case an inspection is required, the DMV is initiated: it publishes another assignment of the same task and an additional (redundant) response is returned by a different worker via the crowdsourcing platform. Based on the two available responses and the historical failure rates of the workers, the probability φ_d of the most probable response d is identified along with the appropriate escalation level ε_d.

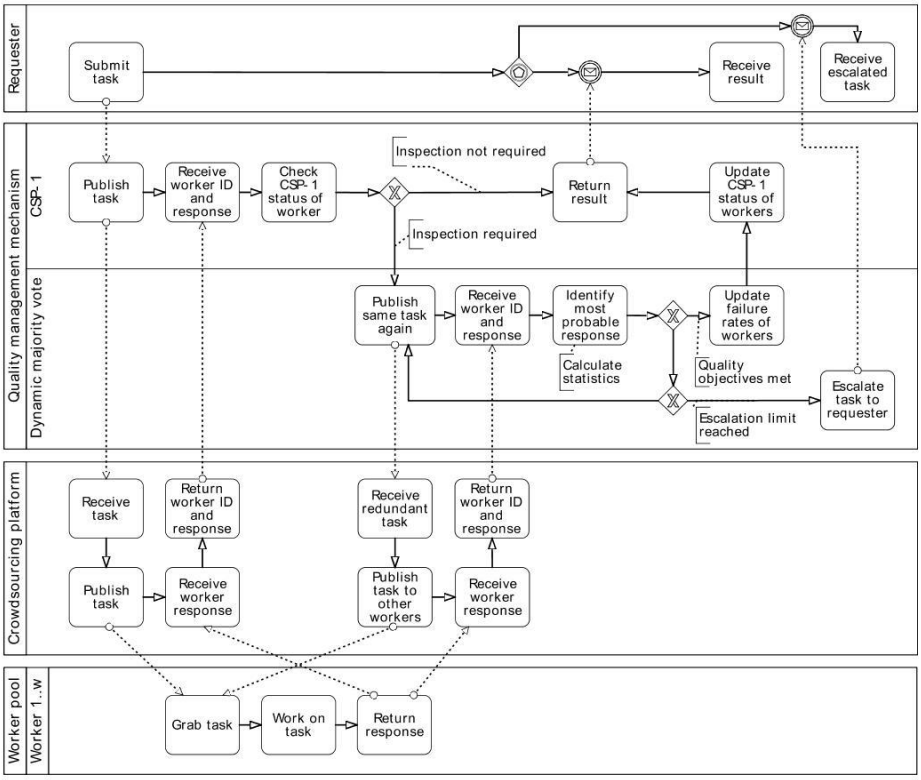

Figure 2: BPMN representation of the CSP/DMV quality management mechanism in the context of the service requester, the crowdsourcing platform and the service workers.

If the probability φ_d exceeds the predefined minimum inspection quality φ_{min}, response d is accepted as the final result and the failure rates of all participating workers are updated depending on whether their response matches d or not. Then, the result is returned to the CSP-1 component where the CSP-1 status indicators of all involved workers is updated as well, i.e. all responses different from d are taken as defects and the CSP-1 status indicators of the appropriate workers are switched to 100% inspection mode. If φ_d is still lower than φ_{min}, additional responses are required from other workers in order to come to a reliable result. Therefore, again another assignment of the task is published, and so on. However, if the escalation level ε_d is falling below the specified escalation limit ε_{min} the loop is interrupted and the task is escalated back to the requester.

3.4 Statistical quality control / CSP-1

The rational for applying statistical quality control to pServices is based on the two assumptions that there is a large number of similar tasks available and

that a worker ID is returned by the crowdsourcing platform with each worker response. The CSP-1 which was introduced in section 2.2 was chosen for the following reasons:

- It supports "real time" pService scenarios in which a continuous stream of tasks need to be completed and for which response time matters.
- It works well with the DMV: The CSP-1 assumes that defective items are replaced, i.e. if an incorrect worker response is detected, it will be replaced by the correct one. The DMV supports that requirement as it implicitly identifies the (most probable) correct result during the sample inspection process.
- Extensions for imperfect sample inspection are available which are required because the DMV only supports a limited inspection quality level φ_{min}.

The assumption of the CSP-1 that the quality of the incoming worker results is under statistical control is addressed in three ways: First, it is assumed that the tasks are similar and require similar skills. Second, an escalation mechanism is provided that identifies outliers. Third, in case sequences of exceptional bad worker results are expected, the CSP-1 may be configured in a way that protects it against them (see section 2.2.1).

For using the CSP/DMV model, the CSP-1 parameters need to be set in the following way:

1. Chose a value for $AOQL$: Set it to the highest average amount of incorrect results that is acceptable for the requester, e.g. a value of $AOQL = 0.05$ means that there is supposed to be a maximum average of 5 percent incorrect results in the result stream.
2. Define an inspection quality level $e_1 = e_2 = \varphi_{min} > AOQL$ that will influence the calculation the CSP-1 parameters as well as the behaviour of the DMV. As a conservative configuration, we recommend to set both, the type 1 and the type 2 inspection error to the minimum inspection quality φ_{min} delivered by the DMV. As a lookahead to section 3.5, the inspection quality level should be rather high in order to allow for a fair assessment of the workers.
3. Specify the incoming level of response quality $\hat{p}$ expected from the workers. We recommend to use the same value as for initializing the worker failure rates in section 3.5.4.
4. Determine i according to section 2.2.1 or select i depending on the typical number of tasks that you expect a worker to complete in a row. For example, it does not make sense to have a clearance number of $i = 20$ if a worker only completes 20 tasks in a row because in that case the worker would be always in the 100% inspection phase.

5. Calculate the corresponding sample fraction f with (2.3) using the values of $AOQL$, e_1, e_2, $\hat{p}$ and i identified above.

3.5 Sample inspection / dynamic majority vote

This chapter provides the statistical foundation for the DMV. After defining a set of terms in section 3.5.1, the model is developed in section 3.5.2 and the resulting equations are presented in 3.5.3. Sections 3.5.4 and 3.5.5 provide complementary information.

3.5.1 Definitions

Let $A = \{a_1, a_2, .., a_n\}$ be the set of all n possible responses to a task. According to the assumptions described in chapter 3.1, exactly one of the possible responses $a \in A$ is the expected correct response for the task. Let

$$\hat{R} = A^w \tag{3.1}$$

be the set of all possible response tuples that might be returned by a group of w different workers who have worked on assignments of the same task. Let $R \in \hat{R}$ be a concrete response tuple with $R = (r_1, r_2, .., r_w)$ and let the tuple $E \in [0;1]^w$ with $E = (p_1, p_2, .., p_w)$ represent the individual historical failure rates $p_1, p_2, .., p_w$ of the contributing workers. The set $D = \{r_1, r_2, .., r_w\} \subseteq A$ is the set of all distinct responses in the response tuple R.

Let $\hat{C} = \{0, 1\}^w$ and $C \in \hat{C}$ be the *correctness profile* of the task. The tuple $C = (c_1, c_2, .., c_w)$ indicates, which worker has returned a correct versus an incorrect response, i.e. $\forall j \leqslant w$:

$$c_j = \begin{cases} 0 & \text{if worker } j \text{ has returned an } \textit{incorrect} \text{ response} \\ 1 & \text{if worker } j \text{ has returned the } \textit{correct} \text{ response}. \end{cases} \tag{3.2}$$

As an example, the correctness profile $(1, 0)$ represents the case that the first of two workers returns a correct response, while the second worker returns an incorrect response.

3.5.2 Statistical considerations

The set $S = \hat{R} \times \hat{C}$ spans the sample space of all possible response tuples $R \in \hat{R}$ and correctness profiles $C \in \hat{C}$. An exemplary sample space for 2 workers is provided by table 1, an example for 3 workers by table 4 (A). The probabilities in each row sum up to the overall probabilities $P_E(R)$ of the response tuples R and the probabilities in each column sum up to the overall probabilities $P_E(C)$ of the correctness profiles. The index E indicates that all probabilities are conditional given the failure rates E of the contributing workers.

Depending on the agreement between the workers, an arbitrary subset of them might return a correct response. As the a priori probability for worker j

Table 1: Sample space of a scenario with 2 workers and 3 possible responses. The rows represent the possible response tuples, the columns the correctness profiles. A historical failure rate of $p = 0.1$ was assumed for both workers.

	Correctness profile C				
Response tuple R	$(0,0)$	$(0,1)$	$(1,0)$	$(1,1)$	$P_E(R)$
(a_1, a_1)	0.0011	0	0	0.2700	0.2711
(a_1, a_2)	0.0011	0.0150	0.0150	0	0.0311
(a_1, a_3)	0.0011	0.0150	0.0150	0	0.0311
(a_2, a_1)	0.0011	0.0150	0.0150	0	0.0311
(a_2, a_2)	0.0011	0	0	0.2700	0.2711
(a_2, a_3)	0.0011	0.0150	0.0150	0	0.0311
(a_3, a_1)	0.0011	0.0150	0.0150	0	0.0311
(a_3, a_2)	0.0011	0.0150	0.0150	0	0.0311
(a_3, a_3)	0.0011	0	0	0.2700	0.2711
$P_E(C)$	0.0100	0.0900	0.0900	0.8100	1.0000

to deliver a correct vs. incorrect response is $q_j = (1 - p_j)$ vs. p_j, the a priori probability $P_E(C)$ for observing a correctness profile C can be estimated as

$$P_E(C) = \prod_{\forall j | c_j = 1} q_j \prod_{\forall j | c_j = 0} p_j \,. \tag{3.3}$$

In order to define the most probable response, the conditional probabilities $P_E(C \mid R)$ of all possible correctness profiles given a specific response tuple R and the tuple of failure rates E are calculated and then the correctness profile $\tilde{C}$ with the highest probability is identified. The estimation is being performed with Bayesian inference using the a priori information about the failure rates of the involved workers and about the distribution of the possible response tuples. $P_E(C \mid R)$ can be calculated as the conditional probability for C given R and E:

$$P_E(C \mid R) = \frac{P_E(C \cap R)}{P_E(R)} = \frac{P_E(R \mid C) P_E(C)}{P_E(R)} \tag{3.4}$$

Within (3.4), the conditional probability $P_E(R \mid C)$ can be calculated as

$$P_E(R \mid C) = \begin{cases} \frac{1}{m_C} & \text{if } \exists k \in D \mid \forall j : (c_j = 1) \Rightarrow (r_j = k), (c_j = 0) \Rightarrow (r_j \neq k) \\ 0 & \text{otherwise} \end{cases} \tag{3.5}$$

with m_C being the number of response tuples R that may be observed given a specific correctness profile C. The underlying assumption is that the a priori probability of all response tuples is equal. For example, if there are three possible responses a_1, a_2 and a_3, there are three response tuples in $\hat{R}$, that apply to the correctness profile $(1, 1)$: (a_1, a_1), (a_2, a_2) and (a_3, a_3). Other response tuples like (a_2, a_3) have a probability of zero, because a_2 and a_3 cannot be correct at the same time as there is only a single correct response $a \in A$.

The number m_C is determined by

$$m_C = \begin{cases} n \cdot \prod\limits_{\forall t | c_t = 0} (n-1) & \text{if } \exists j \mid c_j \neq 0 \\ n^w & \text{if } (\forall j \mid c_j = 0) \wedge (n > w) \\ n^w - \sum\limits_{t=0}^{n} \binom{n}{t} (n-t)^w (-1)^t & \text{if } (\forall j \mid c_j = 0) \wedge (n \leqslant w)\,. \end{cases} \tag{3.6}$$

In the upper branch of (3.6), the response tuple contains the (unknown) correct response $a \in A$ at least once so there are n options to chose it and $(n-1)$ options for choosing each of the incorrect responses. In the middle and lower branch, all responses of the response tuple are incorrect. If there are more possible responses n than workers w, any of the n^w response tuples in $\hat{R}$ might be incorrect. If the number of possible responses is lower than or equal to the number of workers, *Stirling numbers of the second kind* [13] are used to exclude those response tuples that that contain all possible responses out of A. They cannot occur with the the correctness profile $C = (0, 0, .., 0)$ because one of the responses $a \in A$ must be correct.

The marginal probabilities $P(R)$ are determined by

$$P_E(R) = \sum_{\forall C \in \hat{C}} P_E(R \mid C) P_E(C)\,. \tag{3.7}$$

According to (3.5), $P_E(R \mid C)$ and therefore $P_E(C \mid R)$ is different from zero only for such $C_{k,R} \in \hat{C}$ for which all correct results can be mapped to a specific distinct response $k \in D$ and none of the incorrect responses is equal to k. Therefore, (3.4) can be written as:

$$\varphi_k \equiv P_E(C_{k,R} \mid R) = \frac{P_E(C_{k,R})}{m_{C_{k,R}} \cdot P_E(R)} \tag{3.8}$$

with

$$P_E(C_{k,R}) = \prod_{\forall j | r_j = k} q_j \prod_{\forall j | r_j \neq k} p_j\,. \tag{3.9}$$

3.5.3 Final equations

Using (3.5) through (3.9), this turns into the following two equations:

1. If there are more possible responses than workers, i.e. $n > w$, we get

$$\varphi_k = \frac{\prod\limits_{\forall j | r_j = k} q_j \prod\limits_{\forall j | r_j \neq k} p_j}{n \cdot \prod\limits_{\forall j | r_j \neq k} (n-1) \cdot \left(\sum\limits_{k \in D} \frac{\prod\limits_{\forall j | r_j = k} q_j \prod\limits_{\forall j | r_j \neq k} p_j}{n \cdot \prod\limits_{\forall j | r_j \neq k} (n-1)} + \frac{\prod\limits_{\forall j} p_j}{n^w} \right)}\,. \tag{3.10}$$

2. If the number of possible responses is less than or equal to the number of workers, i.e. $n \leqslant w$, we get

$$\varphi_k = \frac{\prod\limits_{\forall j|r_j=k} q_j \prod\limits_{\forall j|r_j\neq k} p_j}{n \cdot \prod\limits_{\forall j|r_j\neq k} (n-1) \cdot \left(\sum\limits_{k\in D} \frac{\prod\limits_{\forall j|r_j=k} q_j \prod\limits_{\forall j|r_j\neq k} p_j}{n \cdot \prod\limits_{\forall j|r_j\neq k} (n-1)} + \frac{\prod\limits_{\forall j} p_j}{n^w - \sum\limits_{t=0}^{n} \binom{n}{t}(n-t)^w(-1)^t} \right)} . \tag{3.11}$$

An example calculation of $\varphi_k = P_E(C_{k,R}, R)$ for the sample space of 2 workers is provided in table 2 and for the sample space of 3 workers in table 5 (A).

Table 2: Conditional probability $P_E(C \mid R)$ for observing a certain correctness profile C given a specific response tuple R in a scenario with 2 workers and 3 possible responses. A historical failure rate of $p = 0.1$ was assumed for both workers.

	$P_E(C \mid R)$ for correctness profile C			
Response tuple R	$(0,0)$	$(0,1)$	$(1,0)$	$(1,1)$
(a_1, a_1)	0.0041	0	0	0.9959
(a_1, a_2)	0.0357	0.4821	0.4821	0
(a_1, a_3)	0.0357	0.4821	0.4821	0
(a_2, a_1)	0.0357	0.4821	0.4821	0
(a_2, a_2)	0.0041	0	0	0.9959
(a_2, a_3)	0.0357	0.4821	0.4821	0
(a_3, a_1)	0.0357	0.4821	0.4821	0
(a_3, a_2)	0.0357	0.4821	0.4821	0
(a_3, a_3)	0.0041	0	0	0.9959

The response $d \in D$ with the highest probability of correctness can finally be calculated as

$$d \in D \mid \forall k \in D : \varphi_d \geqslant \varphi_k \,. \tag{3.12}$$

If the probability φ_d exceeds φ_{min}, the response d is accepted as the correct response. The escalation level is defined as the probability for observing the correctness profile C_d given the failure rates E of the involved workers

$$\varepsilon_d \equiv P_E(C_{d,R}) = \prod_{\forall j|r_j=d} q_j \prod_{\forall j|r_j\neq d} p_j \,. \tag{3.13}$$

Once the probability ε_d underruns ε_{min}, the task is escalated to the requester. The rational of this definition of the escalation level is the following: In general, the quality management mechanism relies on the assumption that the historical failure rates of the workers are good estimates for their future performance. However, there are some tasks which require higher or different skills or which are not solvable at all. For those tasks, the historical worker failure rates are no good estimates and therefore the probability $P_E(C_{d,R})$ of the correctness profile can be much lower than for typical tasks.

3.5.4 Initialization of worker failure rates

For applying the quality management model, initial values for the failure rates p_j of all workers are required. They can be determined by performing a qualification test based on a series of tasks for which the expected results are already known. Only such workers that initially meet a correctness level close to AQQL should be allowed to work on the task by adding them to the worker pool. Workers that don't meet the quality needs would continuously stay in full inspection mode and therefore would lead to high costs.
Note that even if a worker successfully completes all tasks of the qualification test, the initial failure rate must not be set to exactly zero as that would compromise the calculation of φ_k. Because of the nature of human work it is obvious that a failure rate of zero is not a realistic estimate for the future performance of a worker. Therefore, at least one of the test results should be counted as a defect. For example, in a test with 20 tasks, the minimum possible failure rate to achieve would be $p = \frac{1}{20} = 0.05$.

3.5.5 Worker pool management

Depending on the availability of workers, a decision should be made as to which workers are not profitable and should be removed from the worker pool. Therefore, the maximum failure rate p_{max} can be introduced. If a worker's failure rate exceeds the maximum failure rate $p_j > p_{max}$, he may not participate.

4 Architecture and Implementation

Based on the concepts introduced so far we have implemented a prototype system that provides functionalities to manage the quality of tasks on MTurk and similar pService platforms either based on live interaction or simulations. The system considers quality along the two dimensions of correctness and performance. The focus of this paper is on managing the correctness of pService results based on the CSP/DMV approach. Beyond the scope of this paper, we are also experimenting with managing performance by adding time constraints to the optimization problem.

From a software engineering point of view, the quality management system has been designed as a software toolkit that can be utilized to realize a variety of usage scenarios. For instance, the toolkit might act as a software framework to implement third party quality management services that mediate between pService requesters and a pService platform. Another use of the toolkit's software framework would be to enhance given pService client components with the ability of directly managing pService quality. Finally, the toolkit can be used as a software client for pServices as is. In particular, the toolkit allows to run experiments with different quality management approaches and existing or simulated pService platforms in order to produce empirical data.

The general toolkit architecture (Figure 3) is designed for extensibility with respect to quality control mechanisms by means of plugins. It consists of an

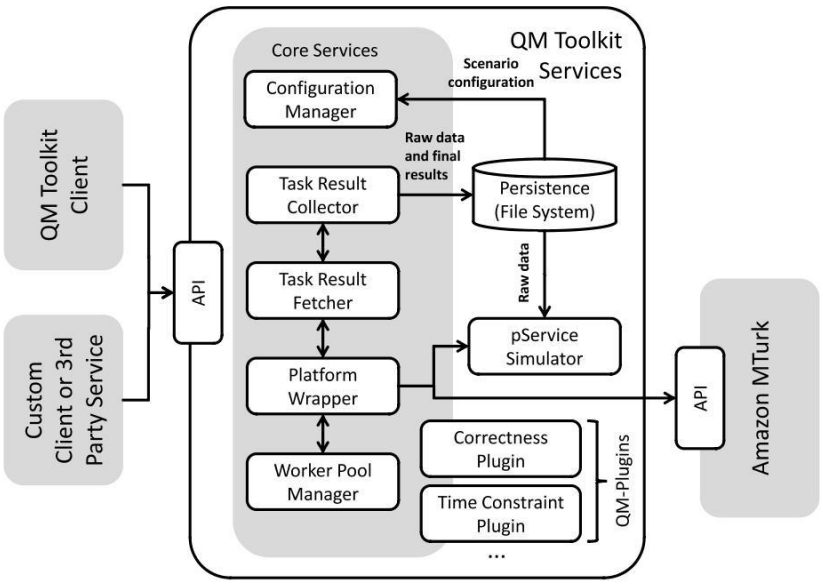

Figure 3: Architectural overview of the pService CSP/DMV toolkit.

infrastructure of *core services* that are used by separate and completely independent quality control plugins. More concretely, the current implementation comprises a *correctness plugin* and a *time constraints plugin*. The toolkit interacts with specific pService platforms by means of an extensible *platform wrapper* component. In our prototype, we have implemented a wrapper for the MTurk platform leveraging its SOAP-based Web service interface for service requesters.

Core services include the *worker pool manager* that is responsible for creating qualification tests and evaluating results. Workers are sorted into pools to control which group may access certain tasks and which may not. This component also provides import and export functionality for worker data and triggers synchronization with the pServices platform so that access rules are reflected there as well and not only locally. The *task result collector* stores worker submissions internally and persists them locally. As correctness control generally requires redundant work, this component also groups redundant submissions for the same task. Plugins may add worker submissions or lock and retrieve them as complete sets. A set of submissions for a task that has been locked may not be accessed by any other component. The *task result fetcher* periodically polls the pServices platform for worker submissions and asserts that a single submission is only processed once. Furthermore, it provides an internal queuing system which may be used by plugins.

A *correctness plugin* generally requires three steps of application logic: (1) retrieve a message from the incoming results queue, (2) process the message, (3) return the data to the task result collector and release the lock. While these steps are supported by the core services, concrete implementations need to refine

them in order to implement a specific management method. Subsequently, our prototype plugin uses CSP-1 and dynamic majority vote for processing the message. It then updates worker statistics via the worker pool manager and returns the results to the task result collector.

The *time constraints plugin* is work in progress that is mentioned to motivate and illustrate our ongoing and future work in this area. The plugin leverages the worker pool manager core service for sorting workers into pools and hence controlling access of workers to tasks. The plugin monitors the progress of task completion and adopts the pool sizes accordingly to complete all tasks at a specified point in time. A trivial solution to complete all tasks by a deadline is to allow all workers to submit results for those tasks. This approach is highly inefficient. Workers who produce poor results in terms of correctness are allowed to participate. Using DMV and CSP-1 does not necessarily decrease total correctness but increases the number of required inspections, thus increasing the total cost. To improve efficiency we allow just a certain percentage of known workers to participate and continuously adapt this fraction based on the progress observed. The goal of the time constraints plugin is to complete all tasks before but also as close as possible to the deadline. For this purpose, the plugin offers a number of basic forecasting algorithms. While we have already found promising experimental results, we will develop a rigorous formal method as part of our future work.

The toolkit was developed using Java SE 6 and is currently maintained as an Eclipse 3.6 project. All components described above are either implemented as separate classes or as a set of classes. Extensibility is generally achieved through usage of abstract classes and interfaces combined with reflections. Regarding the physical design, the toolkit is primarily realized as a library which can be used for multiple purposes, e.g. within a Web application or a standalone tool. Initializer classes can easily be added by adapting the existing standalone initialization class. The current implementation focuses on the MTurk platform and, hence, includes an implementation of the platform wrapper component for this provider. More details on specific implementation aspects have been provided in a separate paper [14].

5 Evaluation

The CSP/DMV model was evaluated on the MTurk platform using an optical character recognition (OCR) scenario. After describing the experimental design in section 5.1, sections 5.2 and 5.3 provide the results of actual experiments.

5.1 Experimental design

The CSP/DMV approach was evaluated using the toolkit described in section 4 by performing an *optical character recognition* (OCR) scenario on top of MTurk. Only such users were allowed to participate who had passed a qualification test. The actual evaluation is divided into two parts: In the first part, the QM

approach was simulated based on a batch of worker responses that had been generated previously. In the second part, the approach was applied in real time.

Data revision, including scenarios such as classifying, tagging, summarizing and revising content or audio and video transcription, and the recognition of (hand) written texts is a class of scenarios which is widely used on Mechanical Turk. Like many data revision tasks, optical character recognition (OCR) of handwritten texts cannot be fully automated yet [15]. Even sophisticated technologies need human assistance in order to achieve satisfying results. The data set consists of 1176 handwritten single words. In each of the tasks, a worker was asked to type in a single handwritten word which was displayed as an image file (JPEG). The expected optimal result (gold standard) was specified by the author of the handwriting himself.

The MTurk platform provides means for limiting the access to tasks to those workers who have successfully completed a so called *qualification test*. Such a test can be designed individually for each type of task. The CSP/DMV approach implicitly determines the failure rates of the workers, therefore there is typically no need to restrict the participation to those who have passed a qualification test. However, a qualification test was used to reduce the overall cost of the experiment as it excludes spammers and workers who submit bad quality right from the start. The test consisted of a series of 10 simple OCR tasks (10 words). In the simulated experiments, the workers had to type in all of them correctly, in the live experiments at least nine of the words had to be correct. According to section 3.5.4 the same initial failure rate of $p = 0.1$ was assumed for all workers who had passed the test.

5.2 Simulation based on a batch of raw results

In order to compare the CSP/DMV approach with the traditional majority vote mechanism, the first part of the evaluation was performed as a simulation on the basis of worker responses from a batch consisting of multiple redundant instances per task. The simulation of the traditional majority vote was performed with a fixed number of raw results. For simulating the approach, a varying number of raw results were used according to the dynamic concept of the approach.

For generating the responses, a batch of 10 assignments per task was uploaded to the MTurk platform on February 1st, 2010. It was prohibited that a worker handles more than one assignment of the same task. The payment was \$0.01 per task, with Amazon receiving a service charge of \$0.005 for each task. Consequently a total amount of $1,176 \times 10 = 11,760$ data sets has been collected during the evaluation leading to total expenses of $11,760 \times$ (\$0.01+\$0.005) =\$176.40.

5.2.1 Execution performance

Probably the most astonishing result of the experiment was the speed with which the worker responses were submitted. In the first pre-tests, a batch of 3,528 tasks was completed by 112 workers in less than 15 minutes at an execution

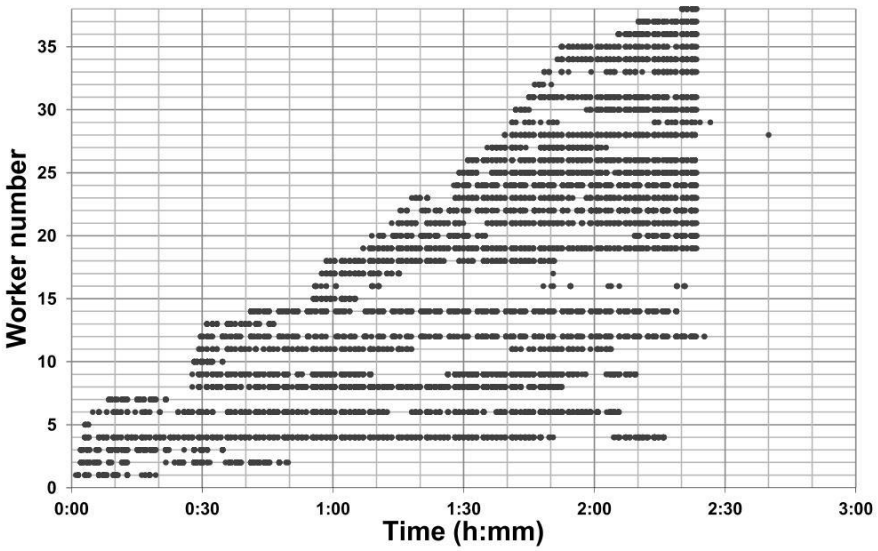

Figure 4: Worker participation during an experiment on MTurk. Each dot represents a single worker response. By and by, additional workers are joining as they finish their work on other types of tasks.

rate of 14,088 tasks per hour. During other experiments we even observed total execution speeds up to 3 times as fast, because of more workers participating. We assume that the execution speed depends on the time of day, since most workers are U.S.- or Indian citizens [16]. Figure 4 illustrates the execution of the actual experiment in which 11,760 tasks have been processed by 38 workers in about 2:30 hours. One can observe how workers successively join the process.

5.2.2 Full inspection

In order to examine the efficiency of the DMV independently from the CSP-1, the first simulation was a full inspection in which the DMV was issued for all tasks, i.e. the CSP-1 was not used at all. Running only the DMV, the specified quality goal of $\varphi_{min} = 0.95$ was even exceeded. Figure 5 illustrates the results of the DMV compared to the traditional MV approach. The traditional MV was simulated based on the same data as the DMV by averaging all possible combinations of 2 to 9 answers within each set of 10 available answers per task for the two-fold up to the 9-fold MV. For each combination, the most occurring answer was chosen. If several answers occurred the same amount of times (tie), a random choice between the answers occurring most was made, as suggested by Snow et al. [17].

The 5th column of table 3 shows that with an accuracy of 0.983, the DMV even outperforms the accuracy of a ninefold traditional MV (0.978). That is a remarkable result given that the DMV is 4 times more efficient as it requires only

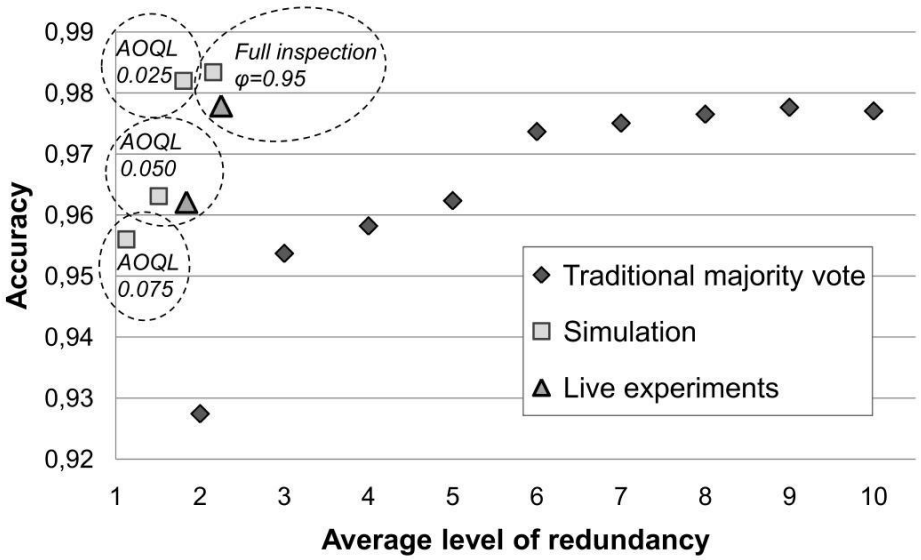

Figure 5: Efficiency of the CSP/DMV approach compared to the traditional majority vote approach. The vertical axis represents the accuracy of the results, the horizontal axis shows the average redundancy i.e. the number workers per task.

2.16 workers per task compared to 9 workers per task for the basic ninefold MV approach. In other words: The DMV approach has reduced the quality management effort by some 75 percent compared to the traditional MV approach. The escalation limit was $\varepsilon_{min} = 0.01$ which caused a number of 29 tasks (2.47%) being escalated. The initial number of possible responses was set to $n = 3$ which is the approximate average number of distinct responses expected in the OCR scenario. If the actual number of distinct responses exceeded 3, n was increased appropriately.

5.2.3 Random inspection with CSP-1

In a series of tests, the CSP/DMV approach has been evaluated for 3 different values of $AOQL$ (2nd through 4th column of table 3). For $AOQL = 0.05$, a total of 1.51 assignments per HIT was observed on average, which is a significant improvement even compared to the 100%-inspection with 2.16 assignments per HIT. 21 HITs were escalated. Some 41 percent of all tasks are inspected. For $AOQL = 0.075$ and even with $AOQL = 0.025$, the quality objectives are again exceeded. However, there are situations where the model does not manage to achieve the desired level anymore. The reason for that lies in the gap between the gold standard and the majority decision of the workers: In several cases, the majority of the workers identified a certain word (e.g. "five") even if the writer (who represented the gold standard) had written a different word (e.g. "fine").

Table 3: Results of the evaluation of the CSP/DMV quality management approach for different types of experiments (simulation versus live) and different quality objectives.

Parameter	Simulations				Live tests	
$AOQL$	0.025	0.050	0.075	*full*	0.050	*full*
i	5	6	1	*inspection*	6	*inspection*
f	0.582	0.233	0.036		0.233	
φ_{min}	0.990	0.990	0.990	0.950	0.990	0.950
ε_{min}	0.010	0.010	0.010	0.010	0.010	0.010
Initial n	3	3	3	3	3	3
Accuracy	0.982	0.963	0.965	0.983	0.962	0.978
AFI	0.664	0.410	0.054	1.000	0.616	1.000
Avg. redundancy	1.800	1.510	1.250	2.157	1.828	2.250
Max. redundancy	5	5	4	4	5	5
Escalated tasks	30	21	19	29	95	44

5.3 Live experiments

For the live experiments, the same tasks were used as for the simulated experiments. However, initially only one assignment of each task was published. Depending on the result returned from the worker and depending on the worker's historical failure rate, the QM approach dynamically decided in real time whether additional assignments had to be published.

5.3.1 Execution performance

In September 2011, two live experiments have been performed on the MTurk platform using the complete data set of 1176 OCR tasks. For the first experiment, the parameters have been the same as in the simulation of the full inspection in section 5.2.2, and for the second experiment the same as in the simulation of the random inspection ($AOQL = 0.05$) described in section 5.2.3. Both, the full inspection as well as the random inspection have resulted in an accuracy comparable to that of the the simulation (table 3). Figure 6 shows the execution performance of the CSP-1 live experiment which has a different characteristic than the batch execution of tasks in figure 4. The number of remaining tasks decreases rather linearly until most of the tasks have been performed. Then, the execution performance slows down dramatically and asymptotically approaches zero. This behavior can be explained by the fact that towards the end, the continuous stream of tasks is interrupted because there are temporarily no tasks available any more. Therefore, the workers consider the process to be completed and go find another task type to work on. However, as the DMV increases the redundancy sequentially, new assignments might be published even after all the available work had already been completed. This applies in particular to complex tasks for which a higher level of redundancy is required because there is less agreement among the workers. The effect is further increased by cherry-picking

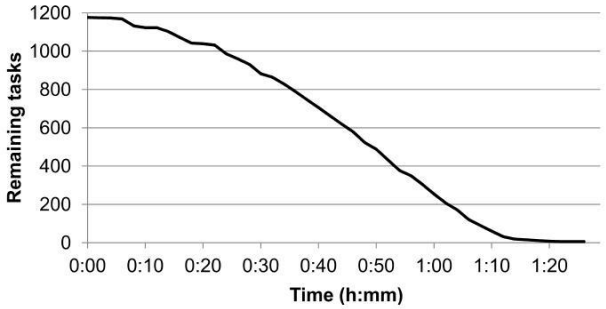

Figure 6: Performance of a live experiment with CSP-1 and DMV: Remaining tasks plotted against the time.

of the workers: Some of the words are so difficult to read that hardly anybody wants to take the risk of making a mistake. In order to avoid discontinuity and speed up the finalization of the process, the QM mechanism should switch to a fixed level of redundancy at the very end of the process. With 61.6% compared to 41.0%, the average fraction inspected (AFI) of the live experiment with the CSP-1 is considerably higher than for the simulation. This difference can be explained with the dynamic nature of the DMV. There is a varying delay before the new assignments are grabbed by the workers and before they return a response. As this delay is not covered by the simulation, it takes more time in the live experiment until the workers build up reputation and until they reach the random inspection phase of the CSP-1. Figure 7 is a modified version of the corresponding one in the original paper [6]. It shows how the AFI changes as the results are being returned to the requester. Note that the horizontal axis is not time but it is the consecutive number of the tasks being returned. At the beginning, the AFI is low and the comparison with figure 6 shows that only a small number of results is being returned in that phase. This is because the process starts with a 100% inspection phase for all workers: In the beginning, at least two assignments are needed before a result can be returned which only shows up in figure 7 after all assignments have been captured. The saw tooth shape of the curve is caused by the alternating i- and f-inspection phases of the workers. The AFI increases with the number of tasks, because more workers are joining the process. In the end, only the most challenging tasks are left over.

6 Related Work

The concept of majority vote is widely used in the context of pServices. Redundant task execution is a basic feature for quality improvement provided by platforms like MTurk. Sorokin and Forsyth as well as Snow et al. have analyzed the effect of the approach based on annotation scenarios [18, 17]. Snow et al. have investigated how many non-experts out of the crowd are needed in order to

Figure 7: Change of the average fraction inspected (AFI)over the task number during the live experiment with $AOQL = 0.05$.

achieve better results than one expert. Depending on the scenario, they report a required number of non-experts between two and more than ten. Whitehill et al. consider how to integrate labeler's expertise into a majority vote mechanism for image labeling [19]. They propose a probabilistic model and use it to simultaneously infer the label of each image, the expertise of each labeler, and the difficulty of each image. Complementary approaches for quality management of pServices include iterative work processes [20], review processes [12] and the injection of gold standard tasks [18]. A maximum likelihood estimation can be used to estimate worker failure rates as well as the correct categories of the task results [21, 12]. The approach leverages the EM algorithm dating back to Dawid and Skene [22]. Raykar et al. propose a specific form of an EM algorithm which is capable of generating a gold standard [23]. A decision matrix was proposed for choosing an adequate QM mechanism depending on the pService scenario [3].

The validity of the majority vote model has been first mathematically proven by Condorcet's Jury Theorem [24]. Under the assumption that one of two outcomes is correct and each decision maker has the independent probability $p > 0.5$ to make the right decision, the probability for a correct group decision is greater than the individual one. Latif-Shabgahi et al. have examined and classified a large number of software voting algorithms used in safety-critical systems [25]. In the field of machine learning, Littlestone and Warmuth developed a dynamic majority algorithm, that acts as a "master algorithm" and aggregates the answers of several prediction algorithms in order to determine the best prediction possible [26]. The aggregation mechanism is a vital part of each majority vote model. Revow et al. compare five combination strategies (majority vote, Bayesian, logistic regression, fuzzy integral, and neural network) and draw the conclusion that majority vote is as effective as the other, more complicated schemes to improve the recognition rate for the data set used [27].

7 Discussion

Crowdsourcing and specifically pServices are a far-reaching subject that might have an extensive impact on the future of employment and human work. Therefore, it needs no be investigated from an interdisciplinary perspective that goes far beyond what we can capture within this paper. Our objective is to help identifying conceptual opportunities and limits that can be utilized as a starting point for a more comprehensive reflection of the topic. With regards to the opportunities, being able to deliver high quality work results is certainly one of the major requirements for a business use of pServices.

However, the proposed CSP/DMV approach goes along with two major restrictions: First, it can only be applied to a subset of scenarios that deal with *deterministic* tasks. Second, it relies on the decision of a majority of workers and cannot come to better results than the group does. Even if the pool of contributing people was carefully selected, there remains a risk for a deliberate or an unintended corruption of work results. For example, if multiple group members are making the same mistake or if they research their responses from the same faulty Wikipedia page, the group decision will be incorrect. Avoiding systematic errors and preventing fraud are therefore certainly two important aspects for further research.

8 Conclusion and Future Work

We have presented an enhanced statistical model for managing the correctness of human-based electronic services (people services) which exploits continuous sampling plans and group decision theory. The model consists of two parts: The continuous sampling plan CSP-1 is used to track the contributions of each worker individually based on samples taken from his or her work results. A *dynamic majority vote* (DMV) approach was introduced for the inspection of the samples which leverages a group decision of multiple workers. The number of workers participating in that group decision is adjusted dynamically depending on their responses and on their individual failure rates. By validating only a fraction of the tasks and keeping the validation effort per task at a minimum, the model is capable of guaranteeing a certain predefined level of result quality at minimum costs. We have implemented our model in form of an extendable software toolkit that acts as a quality management (QM) component for people service platforms. An evaluation on Amazon's Mechanical Turk platform has shown a reduction of the quality management effort of up to 75 percent compared to the traditional majority vote approach.

In our ongoing research we are expanding our QM concepts to other aspects of quality like performance and availability and to more complex use cases, e.g. multi-labeling scenarios. In order to address situations in which the service requester wants to balance the overall benefit with the quality management costs rather than meeting a well defined quality objective at whatever costs, we are investigating the application of a *value of information* (VoI) [28] approach.

References

[1] P. G. Ipeirotis, Analyzing the amazon mechanical turk marketplace, *XRDS* **17** (2010) 16–21.

[2] R. Kern, C. Zirpins and S. Agarwal, Managing quality of Human-Based eServices, in *Service-Oriented Computing - ICSOC 2008 Workshops, ICSOC 2008 International Workshops, Sydney, Australia, December 1st, 2008, Revised Selected Papers*, eds. G. Feuerlicht and W. Lamersdorf *Lecture Notes in Computer Science* **5472**, (Springer, 2009), pp. 304–309.

[3] R. Kern, H. Thies, C. Bauer and G. Satzger, Quality assurance for human-based electronic services: A decision matrix for choosing the right approach, in *Current Trends in Web Engineering*, eds. F. Daniel and F. Facca, *Lecture Notes in Computer Science* **6385** (Springer Berlin/Heidelberg, 2010) pp. 421–424.

[4] D. Montgomery, *Introduction to statistical quality control*, 6th edn. (Wiley & Sons, New York, NY, USA, 2008).

[5] J. Juran and A. Godfrey, *Juran's Quality Handbook*, 5th edn. (McGraw-Hill, New York, NY, USA, 2000).

[6] R. Kern, H. Thies and G. Satzger, Statistical quality control for human-based electronic services, in *Service-Oriented Computing - 8th International Conference, ICSOC 2010, San Francisco, CA, USA, December 7-10, 2010. Proceedings*, eds. P. P. Maglio, M. Weske, J. Yang and M. Fantinato *Lecture Notes in Computer Science* **6470**, (Springer, 2010), pp. 243–257.

[7] H. Dodge and M. Torrey, Additional continuous sampling inspection plans, *Industrial Quality Control* (7) (1951) 7–12.

[8] G. J. Lieberman and H. Solomon, Multi-Level continuous sampling plans, *The Annals of Mathematical Statistics* **26**(4) (1955) 686–704.

[9] D. T. Gosh, An optimum continuous sampling plan CSP-2 with k $\neq$ i to minimise the amount of inspection when incoming quality p follows a distribution, *The Indian Journal of Statistics* **58**(1) (1996) 105–117.

[10] H.-J. Mittag and H. Rinne, *Statistical Methods of Quality Assurance* (Chapman and Hall/CRC, 1993).

[11] R. Wang and C. Chen, Minimum average fraction inspected for continuous sampling plan CSP-1 under inspection error, *Journal of Applied Statistics* **24**(5) (1997) 539–548.

[12] R. Kern, C. Bauer, H. Thies and G. Satzger, Validating results of human-based electronic services leveraging multiple reviewers, in *Proceedings of the 16th Americas Conference on Information Systems (AMCIS)*, (AIS Electronic Library, 2010).

[13] R. L. Graham, D. E. Knuth and O. Patashnik, *Concrete Mathematics: A Foundation for Computer Science*, 2nd edn. (Addison-Wesley Professional, 1994).

[14] D. Bermbach, R. Kern, P. Wichmann, S. Rath, C. Zirpins, D. Bermbach and C. Zirpins, An extendable toolkit for managing quality of human-based electronic services, in *Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[15] D. Lopresti, Optical character recognition errors and their effects on natural language processing, *International Journal on Document Analysis and Recognition (IJDAR)* **12**(3) (2009) 141–151.

[16] J. Ross, L. Irani, M. Silberman, A. Zaldivar and B. Tomlinson, Who are the crowdworkers?: shifting demographics in mechanical turk, in *Proceedings of the 28th of the international conference extended abstracts on Human factors in computing systems*, 2010, pp. 2863–2872.

[17] R. Snow, B. O'Connor, D. Jurafsky and A. Y. Ng, Cheap and fast—but is it good? evaluating non-expert annotations for natural language tasks, in *EMNLP'08: Proceedings of the Conference on Empirical Methods in Natural Language Processing*, (ACL, Stroudsburg, USA, 2008), pp. 254–263.

[18] A. Sorokin and D. Forsyth, Utility data annotation with amazon mechanical turk, in *CVPRW '08: Proceedings of the Conference on Computer Vision and Pattern Recognition Workshops*, (IEEE Computer Society, Washington, WA, USA, June 2008), pp. 1–8.

[19] J. Whitehill, P. Ruvolo, T. Wu, J. Bergsma and J. Movellan, Whose vote should count more: Optimal integration of labels from labelers of unknown expertise, in *Advances in Neural Information Processing Systems 22*, 2009, pp. 2035–2043.

[20] G. Little, L. B. Chilton, M. Goldman and R. C. Miller, Turkit: human computation algorithms on mechanical turk, in *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, *UIST '10*, (ACM, New York, NY, USA, 2010), pp. 57–66.

[21] P. Ipeirotis, F. Provost and J. Wang, Quality management on amazon mechanical turk, in *Proceedings of the ACM SIGKDD Workshop on Human Computation*, (ACM, New York, NY, USA, 2010), pp. 64–67.

[22] A. Dawid and A. Skene, Maximum likelihood estimation of observer Error-Rates using the EM algorithm, *Journal of the Royal Statistical Society* **28**(1) (1979) 20–28.

[23] V. Raykar, S. Yu, L. Zhao, G. Valadez, C. Florin, L. Bogoni and L. Moy, Learning from crowds, *The Journal of Machine Learning Research* **99** (2010) 1297–1322.

[24] J. de A. Condorcet, *Essai sur l'application de l'analyse a la probabilite des decisions rendues a la pluralite des voix*, 1st edition edn. (Imprimerie royale, Paris, France, 1785).

[25] G. Latif-Shabgahi, J. Bass and S. Bennett, A taxonomy for software voting algorithms used in safety-critical systems, *Reliability, IEEE Transactions on* **53**(3) (2004) 319–328.

[26] N. Littlestone and M. K. Warmuth, The weighted majority algorithm, *Information and Comutation* **108** (1994) 212–261.

[27] M. Revow, C. K. I. Williams and G. E. Hinton, Using generative models for handwritten digit recognition, *IEEE Trans. Pattern Anal. Mach. Intell.* **18**(6) (1996) 592–606.

[28] J. Marschak and R. Radner, Economic Theory of Teams. Chapter 1, Cowles Foundation Discussion Papers 59a, Cowles Foundation for Research in Economics, Yale University (1958).

A Auxiliary tables

Table 4: Sample space of a scenario with 3 workers and 3 possible responses. A historical failure rate of $p = 0.1$ was assumed for all workers.

#	Result set R	Correctness profile C								
		(0,0,0)	(0,0,1)	(0,1,0)	(0,1,1)	(1,0,0)	(1,0,1)	(1,1,0)	(1,1,1)	$P_E(R)$
1	$a_1a_1a_1$	0.00005	0	0	0	0	0	0	0.24300	0.24305
2	$a_1a_1a_2$	0.00005	0.00075	0	0	0	0	0.01350	0	0.01430
3	$a_1a_1a_3$	0.00005	0.00075	0	0	0	0	0.01350	0	0.01430
4	$a_1a_2a_1$	0.00005	0	0.00075	0	0	0.01350	0	0	0.01430
5	$a_1a_2a_2$	0.00005	0	0	0.01350	0.00075	0	0	0	0.01430
6	$a_1a_2a_3$	0	0.00075	0.00075	0	0.00075	0	0	0	0.00225
7	$a_1a_3a_1$	0.00005	0	0.00075	0	0	0.01350	0	0	0.01430
8	$a_1a_3a_2$	0	0.00075	0.00075	0	0.00075	0	0	0	0.00225
9	$a_1a_3a_3$	0.00005	0	0	0.01350	0.00075	0	0	0	0.01430
10	$a_2a_1a_1$	0.00005	0	0	0.01350	0.00075	0	0	0	0.01430
11	$a_2a_1a_2$	0.00005	0	0.00075	0	0	0.01350	0	0	0.01430
12	$a_2a_1a_3$	0	0.00075	0.00075	0	0.00075	0	0	0	0.00225
13	$a_2a_2a_1$	0.00005	0.00075	0	0	0	0	0.01350	0	0.01430
14	$a_2a_2a_2$	0.00005	0	0	0	0	0	0	0.24300	0.24305
15	$a_2a_2a_3$	0.00005	0.00075	0	0	0	0	0.01350	0	0.01430
16	$a_2a_3a_1$	0	0.00075	0.00075	0	0.00075	0	0	0	0.00225
17	$a_2a_3a_2$	0.00005	0	0.00075	0	0	0.01350	0	0	0.01430
18	$a_2a_3a_3$	0.00005	0	0	0.01350	0.00075	0	0	0	0.01430
19	$a_3a_1a_1$	0.00005	0	0	0.01350	0.00075	0	0	0	0.01430
20	$a_3a_1a_2$	0	0.00075	0.00075	0	0.00075	0	0	0	0.00225
21	$a_3a_1a_3$	0.00005	0	0.00075	0	0	0.01350	0	0	0.01430
22	$a_3a_2a_1$	0	0.00075	0.00075	0	0.00075	0	0	0	0.00225
23	$a_3a_2a_2$	0.00005	0	0	0.01350	0.00075	0	0	0	0.01430
24	$a_3a_2a_3$	0.00005	0	0.00075	0	0	0.01350	0	0	0.01430
25	$a_3a_3a_1$	0.00005	0.00075	0	0	0	0	0.01350	0	0.01430
26	$a_3a_3a_2$	0.00005	0.00075	0	0	0	0	0.01350	0	0.01430
27	$a_3a_3a_3$	0.00005	0	0	0	0	0	0	0.24300	0.24305
	$P_E(C)$	0.00100	0.00900	0.00900	0.08100	0.00900	0.08100	0.08100	0.72900	1.00000

Table 5: Conditional probability $P(C \mid R)$ for observing a certain correctness profile C given specific response tuple R for the sample space in table 4.

#	Result set R	Conditional probability $P_E(C \mid R)$ for correctness profile C							
		(0,0,0)	(0,0,1)	(0,1,0)	(0,1,1)	(1,0,0)	(1,0,1)	(1,1,0)	(1,1,1)
1	$a_1a_1a_1$	0.0002	0	0	0	0	0	0	0.9998
2	$a_1a_1a_2$	0.00333	0.05246	0	0	0	0	0.94421	0
3	$a_1a_1a_3$	0.00333	0.05246	0	0	0	0	0.94421	0
4	$a_1a_2a_1$	0.00333	0	0.05246	0	0	0.94421	0	0
5	$a_1a_2a_2$	0.00333	0	0	0.94421	0.05246	0	0	0
6	$a_1a_2a_3$	0	0.33333	0.33333	0	0.33333	0	0	0
7	$a_1a_3a_1$	0.00333	0	0.05246	0	0	0.94421	0	0
8	$a_1a_3a_2$	0	0.33333	0.33333	0	0.33333	0	0	0
9	$a_1a_3a_3$	0.00333	0	0	0.94421	0.05246	0	0	0
10	$a_2a_1a_1$	0.00333	0	0	0.94421	0.05246	0	0	0
11	$a_2a_1a_2$	0.00333	0	0.05246	0	0	0.94421	0	0
12	$a_2a_1a_3$	0	0.33333	0.33333	0	0.33333	0	0	0
13	$a_2a_2a_1$	0.00333	0.05246	0	0	0	0	0.94421	0
14	$a_2a_2a_2$	0.0002	0	0	0	0	0	0	0.9998
15	$a_2a_2a_3$	0.00333	0.05246	0	0	0	0	0.94421	0
16	$a_2a_3a_1$	0	0.33333	0.33333	0	0.33333	0	0	0
17	$a_2a_3a_2$	0.00333	0	0.05246	0	0	0.94421	0	0
18	$a_2a_3a_3$	0.00333	0	0	0.94421	0.05246	0	0	0
19	$a_3a_1a_1$	0.00333	0	0	0.94421	0.05246	0	0	0
20	$a_3a_1a_2$	0	0.33333	0.33333	0	0.33333	0	0	0
21	$a_3a_1a_3$	0.00333	0	0.05246	0	0	0.94421	0	0
22	$a_3a_2a_1$	0	0.33333	0.33333	0	0.33333	0	0	0
23	$a_3a_2a_2$	0.00333	0	0	0.94421	0.05246	0	0	0
24	$a_3a_2a_3$	0.00333	0	0.05246	0	0	0.94421	0	0
25	$a_3a_3a_1$	0.00333	0.05246	0	0	0	0	0.94421	0
26	$a_3a_3a_2$	0.00333	0.05246	0	0	0	0	0.94421	0
27	$a_3a_3a_3$	0.0002	0	0	0	0	0	0	0.9998