Context-Aware Worker Selection For Efficient Quality Control In Crowdsourcing

PhD thesis submitted in cotutelle-procedure to :

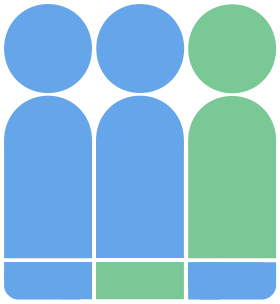
| Institut national des sciences appliquées de Lyon |

Ecole doctorale d'Informatique et de Mathématiques (InfoMaths)

&

| Universität Passau |

Fakultät für Informatik und Mathematik



Tarek AWWAD

| Supervisors |

Prof. Dr. Lionel BRUNIE

Prof. Dr. Harald KOSCH

| Co-Supervisors |

Dr. Nadia BENNANI

Dr. Veronika REHN-SONIGO

November 30, 2018

“Ez ikusi, Ez ikasi ...” - *Antoine d’Abbadie*

Abstract

In the last decade, crowdsourcing has proved its ability to address large scale data collection tasks, such as labeling large data sets, at a low cost and in a short time. However, the performance and behavior variability between workers as well as the variability in task designs and contents, induce an unevenness in the quality of the produced contributions and, thus, in the final output quality. In order to maintain the effectiveness of crowdsourcing, it is crucial to control the quality of the contributions. Furthermore, maintaining the efficiency of crowdsourcing requires the time and cost overhead related to the quality control to be at its lowest. While effective, current quality control techniques such as contribution aggregation, worker selection, context-specific reputation systems, and multi-step workflows, suffer from fairly high time and budget overheads and from their dependency on prior knowledge about individual workers.

In this thesis, we address this challenge by leveraging the similarity between completed and incoming tasks as well as the correlation between the worker declarative profiles and their performance in previous tasks in order to perform an efficient task-aware worker selection. To this end, we propose CAWS (Context Aware Worker Selection) method which operates in two phases; in an *offline* phase, completed tasks are clustered into homogeneous groups for each of which the correlation with the workers declarative profile is learned. Then, in the *online* phase, incoming tasks are matched to one of the existing clusters and the correspondent, previously inferred profile model is used to select the most reliable online workers for the given task. Using declarative profiles helps eliminate any probing process, which reduces the time and the budget while maintaining the crowdsourcing quality. Furthermore, the set of completed tasks, when compared to a probing task split, provides a larger corpus from which a more precise profile model can be learned. This translates to a better selection quality, especially for harder tasks.

In order to evaluate CAWS, we introduce CrowdED (Crowdsourcing Evaluation Dataset), a rich dataset to evaluate quality control methods and quality-driven task vectorization and clustering. The generation of CrowdED relies on a constrained sampling approach that allows to produce a task corpus which respects both, the budget and type constraints. Beside helping in evaluating CAWS, and through its generality and richness, CrowdED helps in plugging the benchmarking gap present in the crowdsourcing quality control community.

Using CrowdED, we evaluate the performance of CAWS in terms of the quality of the worker selection and in terms of the achieved time and budget reduction. Results shows the following: first, automatic grouping is able to achieve a learning quality similar to job-based grouping. And second, CAWS is able to outperform the state-of-the-art profile-based worker selection when it comes to quality. This is especially true when strong budget and time constraints are present on the requester side.

Finally, we complement our work by a software contribution consisting of an open source framework called CREX (CReate Enrich eXtend). CREX allows the creation, the extension and the enrichment of crowdsourcing datasets. It provides the tools to vectorize, cluster and sample a task corpus to produce constrained task sets and to automatically generate custom crowdsourcing campaign sites.

Résumé

Le crowdsourcing est une technique rapide, efficace et peu onéreuse destinée à recueillir les opinions d'un large public (dont les membres sont appelés *workers*) sur un ensemble de questions (se présentant sous forme de *tâches*) et permettant ainsi à des utilisateurs (appelés *requesters*) d'obtenir un consensus de réponse ou d'expérimenter un ressenti. La disparité comportementale et de performances des workers d'une part et la variété en termes de contenu et de présentation des tâches par ailleurs influent considérablement sur la qualité des contributions recueillies et par conséquent sur le résultat finalement obtenu par les techniques d'agrégation appliquées sur l'ensemble des contributions. Par conséquent, garder leur légitimité impose aux plateformes de crowdsourcing de se doter de mécanismes permettant l'obtention de réponses fiables et de qualité dans un délai et avec un budget optimisé. Les techniques actuellement proposées (à savoir, les solutions visant à améliorer les résultats agrégés, la sélection des workers, ... etc) bien que efficaces, n'optimisent pas les délais de réponse et le coût des campagnes et se basent sur une connaissance préalable des workers.

Dans cette thèse, nous proposons CAWS (Context AwareWorker Selection), une méthode de contrôle de la qualité des contributions dans le crowdsourcing visant à optimiser le délai de réponse et le coût des campagnes. CAWS exploite la similarité entre les tâches déjà traitées et celles à traiter d'une part ainsi que la corrélation existant entre le profil des utilisateurs et les performances de leurs réponses aux tâches préalablement traitées. Pour ce faire, CAWS se compose de deux phases, une phase d'apprentissage opérant hors-ligne et pendant laquelle les tâches de l'historique sont regroupées de manière homogène sous forme de clusters. Pour chaque cluster, un profil type optimisant la qualité des réponses aux tâches le composant, est inféré ; la seconde phase permet à l'arrivée d'une nouvelle tâche de sélectionner les meilleurs workers connectés pour y répondre. Il s'agit des workers dont le profil présente une forte similarité avec le profil type du cluster de tâches, duquel la tâche nouvellement créée est la plus proche. La recherche de workers en se basant uniquement sur la similarité de profil rend ainsi inutile de tester en ligne les workers connectés et réduit ainsi les coûts en temps et en budget de la campagne tout en maintenant la même qualité des réponses. De plus, de par sa taille, l'historique des tâches permet une inférence plus aisée du profil des utilisateurs requis pour un type de tâche, ce qui permet à CAWS de donner de meilleures performances dans le cas de tâches difficiles.

La seconde contribution de la thèse est de proposer un jeu de données, appelé CrowdED (Crowdsourcing Evaluation Dataset), ayant les propriétés requises pour, d'une part, tester les performances de CAWS et les comparer aux méthodes concurrentes et d'autre part, pour tester et comparer l'impact des différentes méthodes de catégorisation des tâches de l'historique (c'est-à-dire, la méthode de vectorisation des tâches et l'algorithme de clustering utilisé) sur la qualité du résultat, tout en utilisant un jeu de tâches unique (obtenu par une

méthode d'échantillonnage), respectant les contraintes budgétaires et gardant les propriétés de validité en terme de dimension. Par ailleurs, CrowdED rend possible la comparaison de méthodes de contrôle de qualité quelle que soient leurs catégories, du fait du respect d'un cahier des charges lors de sa constitution.

Les résultats de l'évaluation de CAWS en utilisant CrowdED comparés aux méthodes concurrentes basées sur la sélection de workers, donnent des résultats meilleurs, surtout en cas de contraintes temporelles et budgétaires fortes. Les expérimentations réalisées avec un historique structuré en catégories donnent des résultats comparables à des jeux de données où les taches sont volontairement regroupées de manière homogène. La dernière contribution de la thèse est un outil appelé CREX (CReate Enrich eXtend) dont le rôle est de permettre la création, l'extension ou l'enrichissement de jeux de données destinés à tester des méthodes de crowdsourcing. Il propose des modules extensibles de vectorisation, de clusterisation et d'échantillonnages et permet une génération automatique d'une campagne de crowdsourcing.

Zusammenfassung

Im letzten Jahrzehnt hat Crowdsourcing seine Fähigkeit bewiesen große Datensammelaufgaben, wie die Beschriftung großer Datensätze, zu geringen Kosten und in kurzer Zeit zu bewältigen. Die Leistungs- und Verhaltensschwankungen zwischen den Arbeitern sowie die Variabilität in den Aufgabenentwürfen und -inhalten führen jedoch zu einer Ungleichmäßigkeit in der Qualität der erworbenen Beiträge und somit in der endgültigen Ausgabequalität. Um die Effektivität von Crowdsourcing zu erhalten, ist es entscheidend die Qualität der einzelnen Beiträge zu kontrollieren. Darüber hinaus erfordert die Aufrechterhaltung der Effizienz von Crowdsourcing, dass der Zeit- und Kostenaufwand für die Qualitätskontrolle am geringsten ist. Effektive, aktuelle Qualitätskontrolltechniken wie die Aggregation von Beiträgen, die gezielte Auswahl von Arbeitern, kontextspezifische Reputationssysteme und mehrstufige Workflows leiden unter ziemlich hohen Zeit- und Budgetzwangslagen und von ihrer Abhängigkeit von vorausgehenden Kenntnissen über die einzelnen Arbeiter.

Ìn dieser Arbeit gehen wir diese Herausforderungen an, indem wir die Ähnlichkeit zwischen abgeschlossenen und eingehenden Aufgaben sowie die Korrelation zwischen den von Arbeitern deklarierten Profilen und deren Leistung in früheren Aufgaben nutzen, um eine effiziente aufgabenbewusste Arbeiterauswahl durchzuführen. Zu diesem Zweck schlagen wir eine zweiphasige Methode vor: CAWS (Context Aware Worker Selection). In einer Offline-Phase werden bereits bearbeitete Aufgaben in homogene Cluster gruppiert, für welche jeweils die Korrelation mit dem vorab deklarierten Profil der Arbeiter erlernt wird. In der Online-Phase werden eingehende Aufgaben dann einem der vorhandenen Cluster zugeordnet, und das entsprechende, zuvor erschlossene Profilmodell wird dazu verwendet, um die vertrauenswürdigsten Online-Mitarbeiter für die gegebene Aufgabe auszuwählen. Die Verwendung von deklarativen Profilen hilft dabei jeglichen Sondierungsprozess zu eliminieren, wobei Zeit und Kosten reduziert werden und gleichzeitig die Crowdsourcing-Qualität beibehalten wird. Darüber hinaus bietet das Aggregat der abgeschlossenen Aufgaben im Vergleich zu einer Aufgabenaufteilung durch Sondierung einen größeren Korpus, aus dem ein präziseres Profilmodell erlernt werden kann. Dies führt zu einer besseren Auswahlqualität, insbesondere für schwierigere Aufgaben.

Um CAWS zu evaluieren, stellen wir CrowdED (Crowdsourcing Evaluation Dataset) vor, einen umfassenden Datensatz zur Evaluierung von Qualitätskontrollmethoden und qualitätsgetriebener Aufgaben-Vektorisierung und Clusterbildung. Die Generierung von CrowdED basiert auf einem bedingten Stichprobeverfahren, welches es ermöglicht, einen Aufgaben-Corpus zu erstellen, der sowohl die Budget- als auch die Typ-Bedingungen einhält. Neben seiner Allgemeingültigkeit und Reichhaltigkeit, hilft CrowdED nicht nur bei der Bewertung von CAWS, sondern es hilft auch dabei, die Benchmarking-Lücke in der Crowdsourcing-Community für Qualitätskontrolle zu schließen.

Mit CrowdED evaluieren wir die Leistung von CAWS im Hinblick auf die Qualität der Arbeiterauswahl und auf die erreichte Zeit- und Kostenreduzierung. Die Ergebnisse zeigen folgendes: Zum einen kann mit der automatischen Gruppierung eine Lernqualität ähnlich der von Job-basierten Gruppierungen erreicht werden. Und zweitens ist CAWS in der Lage, die aktuellen profilbasierten Auswahlmethoden in Bezug auf Qualität zu übertreffen. Dies gilt insbesondere dann, wenn auf der Anfordererseite starke Budget- und Zeitbeschränkungen bestehen.

Schließlich ergänzen wir unsere Arbeit mit einer Software, die aus einem lizenzfreien Framework namens CREX (CReate Enrich eXtend) besteht. CREX ermöglicht die Erstellung, Erweiterung und Anreicherung von Crowdsourcing-Datensätzen. Es liefert die nötigen Werkzeuge um einen Aufgabenkorpus zu vektorisieren, zu gruppieren und zu samplen, um eingeschränkte Aufgabensätze zu erzeugen und um automatisch benutzerdefinierte Crowdsourcing-Kampagnen-Seiten zu generieren.

Contents

Abstract iii

Résumé v

Zusammenfassung vii

Contents ix

List of Figures xiii

List of Tables xvii

1 Introduction 1

1.1 Context **1**

1.1.1 Surfing the crowd 1

1.1.2 Crowdsourcing: a definition 2

1.1.3 Crowdsourcing in use 4

1.2 The quality issue in crowdsourcing systems **5**

1.3 Dealing with the quality issue in crowdsourcing systems **6**

1.4 Research challenges and thesis contributions **7**

1.5 Thesis outline **9**

2 State of the art . . . 11
2.1 Aggregation optimization approaches 12
2.1.1 Non-iterative aggregation . . . 12
2.1.2 Iterative aggregation . . . 13
2.2 Worker selection approaches 15
2.2.1 Worker screening . . . 15
2.2.2 Worker tracking . . . 16
2.2.3 Skill matching . . . 17
2.2.4 Profile based selection . . . 18
2.2.5 Preference based assignment . . . 19
2.3 Individual performance enhancement 20
2.3.1 Incentives . . . 20
2.3.2 Training . . . 21
2.3.3 Task interface design and Priming . . . 22
2.4 Multiple stage crowdsourcing 22
2.5 Thesis positioning 23
2.5.1 On the temporal aspect of quality control . . . 24
2.5.2 The impact of worker selection on the aggregation process . . . 26
2.5.3 Discussing worker selection strategies . . . 30
2.5.4 Positioning summary . . . 32
2.6 Summary 33
3 CAWS : a method for Context Aware Worker Selection . . . 35
3.1 Setup and problem formalization 36
3.1.1 System components . . . 36
3.1.2 Task and worker characterization . . . 38
3.1.3 The worker selection problem . . . 39
3.2 Context Aware Worker Selection 40
3.2.1 Vision . . . 40
3.2.2 Method overview . . . 41
3.3 Offline learning 42
3.3.1 Task Grouping . . . 42
3.3.2 The Discovery Algorithm . . . 45
3.4 Online crowdsourcing 47
3.5 Summary 48
4 CrowdED and CREX : Enabling Quality Control Evaluation . . . 51
4.1 Objectives and requirements 52
4.2 Crowdsourcing evaluation datasets 54
4.2.1 An overview of the state-of-the-art . . . 54
4.2.2 The benchmarking gap . . . 58

4.3 CrowdED : Crowdsourcing Evaluation Dataset 59
4.3.1 Data preparation . . . 60
4.3.2 Data collection . . . 67
4.3.3 Data Structure and statistics . . . 69
4.4 CREX: CReate Enrich eXtend 74
4.4.1 Data preparation component (CREX-D) . . . 74
4.4.2 Campaign management component (CREX-C) . . . 77
4.5 CrowdED and CREX re-usability 79
4.5.1 Usability in quality control evaluation . . . 79
4.5.2 Compliance with the FAIR principals . . . 79
4.6 Summary 80

5 Evaluating the Context Aware Worker Selection Method . . . 83
5.1 Evaluation metrics 83
5.1.1 Quality metrics . . . 83
5.1.2 Overhead measurement . . . 84
5.2 General experimental setup 85
5.2.1 Dataset . . . 85
5.2.2 Ground truth . . . 86
5.2.3 Selection methods . . . 86
5.2.4 Task grouping . . . 86
5.2.5 Aggregation . . . 87
5.3 Experiments 88
5.3.1 Evaluating the performance of CAWS . . . 88
5.3.2 Evaluating the time and the budget gain . . . 94
5.4 Summary 98

6 Conclusion and perspectives . . . 101
6.1 Conclusion 101
6.2 Perspectives 102
6.2.1 Feature Taxonomy . . . 103
6.2.2 Adding worker preference . . . 105
6.2.3 Concept shift and model adjustment . . . 105
6.2.4 Quality guarantees . . . 105

Appendices . . . 107

A Details about the CREX platform . . . 107
A.1 Dependencies 107
A.2 References 107
A.3 Configurations 107

References . . . 109

List of Figures

1.1 Literature usage trend for crowd computing related keywords between 2005 and 2017. . . . 2
1.2 A typical workflow of a corwdsourcing campaign. . . . 4

2.1 The workflow of the EM aggregation where $\theta[it]$ represent the model parameters to estimate e.g., worker accuracy scores. . . . 14
2.2 A projection of different quality control approaches on the crowdsourcing time line. . . . 25
2.3 The beta distributions used to simulate the worker accuracy distributions with various configurations. . . . 26
2.4 The worker sampling process for *Experiment I*. . . . 27
2.5 The average accuracy of EM in terms of the ratio of reliable workers in the crowd for various worker accuracy distributions. . . . 28
2.6 A comparison of the behavior of three different aggregation techniques for various distributions of reliable worker (x-axis) in the targeted group of size 20. 28
2.7 The worker sampling process for *Experiment II*. . . . 30
2.8 A comparison of the behavior of three different aggregation techniques with respect to the number of workers for three different configurations: bad workers, random workers and good workers. . . . 31

3.1 The structure of a crowdsourcing job and task. . . . 37
3.2 An overview on the workflow of CAWS. . . . 42
3.3 An overview of the offline learning phase of CAWS. . . . 43
3.4 An illustrative example of the grouping importance. . . . 44

3.5 An illustrative example of the behavior of different clustering algorithms (columns) w.r.t various data point distribution (rows). 45
3.6 An overview of the online crowdsourcing phase of CAWS. 47

4.1 An overview of the creation process of CrowdED. 60
4.2 An instantiation of the constrained sampling problem. Here, two clusterings systems of two clusters each are considered. The maximum sample size *S* and the minimum sample-per-cluster size is *th*. 62
4.3 An overview of the iterative process of the random search. 67
4.4 An overview of the structural characteristics of CrowdED. (*) a dense contribution is a set of answers given by a single worker to the entire task set. 69
4.5 The distribution of workers over a set of demographical features: (a) age, (b) gender, (c)native language and (d) country. 71
4.6 The distribution of workers over a set of knowledge related features: (a) education level, (b) work experience, (c) education domain and (d) work domain. 72
4.7 The distribution of workers over a set of interest related features: (a) and (b) interests and (c) other spoken language. 73
4.8 The distribution of the self evaluation rating for different knowledge domains w.r.t. the gender of the worker 74
4.9 The description of the profile rating task. 75
4.10 (An overview of the CREX framework that combines two main components: **CREX-D** for data selection and **CREX-C** for campaign generation and data collection. 76
4.11 A summarized view of the CREX module configurable parameters. 76
4.12 Overview of the CREX-C component. 77
4.13 The relation between the quality of the profile (in terms of consistency) and the time spend to fill it : (a) Average rating, (b) Median rating and (c) Maximum rating 79

5.1 The accuracy of (a) MV and (b) EM w.r.t. the number of selected workers for CAWS - LR and CAWS - RF 89
5.2 A comparison of the accuracy two aggregation algorithms EM and MV w.r.t. the number of the workers selected by CAWS. 90
5.3 Accuracy of the aggregation process using MV w.r.t. the number of workers selected by CAWS, compared to the accuracy for: Li with 5 probing tasks, Li with 10 probing tasks and a random selection process. 91
5.4 The accuracy of the aggregation process applied on the contributions of workers selected by CAWS, Li and random for individual subsets of the dataset: (a) DE0 and (b) BI0. 92
5.5 The accuracy of the aggregation process applied on the contributions of workers selected by CAWS, Li and random for individual subsets of the dataset: (a) A80, (b) CH0. 93

5.6 A comparison of the accuracy of the MV aggregation processes w.r.t. the number of the workers selected by CAWS for different clustering combinations and a set based grouping. 94
5.7 A comparison of the aggregation accuracy processes w.r.t. the number of the selected workers by CAWS (clustering) and by Li. 95
5.8 Time and budget overheads. 97
5.9 EM accuracy for a fixed budget: (a) $\lambda = 21$ and (b) $\lambda = 35$ (Exp.6) 98

List of Tables

2.1 A comparison of the worker selection approaches in terms of the task specificity, worker specificity and a priori dependency. 33

3.1 A summarized view of the annotations used in this chapter. 40

4.1 A comparison of a sample of dataset used in the literature to evaluate crowdsourcing quality control. 57
4.2 The needs of selected quality control methods in terms of dataset content 59
4.3 On overview of the task sets used to build initial task corpus of CrowdED. . . . 61
4.4 The sampling problem constraints. 65
4.5 The usability of CrowdED w.r.t the existing quality control methods 80

5.1 A summarized view of the selection methods used in our evaluations. 86
5.2 A summarized view of the grouping methods used in our evaluations. 87

1

Introduction

1.1 Context

1.1.1 Surfing the crowd

Throughout the early and pre-modern human history, group effort was harnessed to accomplish daily tasks like group foraging, edifice building and military conquests with a higher efficiency and at a lower risk. Since the late modern period (∼mid-18th century) and with the emergence of its lifestyle principals and models such as the democracy, the industry and the economy, the limits of group effort have been stretched out to include a more cognitive aspect of the human skills, that is the "group opinion". Voting, consensus and other forms of the group opinion have proved their ability to lead an efficient decision making process and to produce a high-confidence output even in complex situations and contexts. Perfect examples of this are elections and referendums in politics as well as brainstorming and user experience tallying in the world of economics, industry and services. With the advent of the Web, the domains where the group opinion is sought as well as the size and diversity of the reachable crowd have both exploded. This allowed for an easier and wider exploitation of the group effort. Progressively, a new generation of web-enabled crowd-centric and crowd-supported platforms and services has emerged such as social media platforms (e.g., Facebook [30] and Twitter [124]), opinion-sharing sites like blogs and forums (e.g., Reddit[103] and Quora[95]) as well as platforms for co-creation (e.g., Wikipedia[133] and Thingiverse[119]), for crowdfunding (e.g., Kickstarter[64] and Indegogo[50]), for crowd sensing (e.g., Apisense[4] and Kumuluz[67]) and for micro-tasks solving e.g., AMT[123] and Figure Eight[32]). In the literature, these systems are referred to, globally or partially, using many terms such as *crowd computing*[10, 86, 93, 114], *human computation*, *collective intelligence* [94], etc. In this work we will refer to these systems using the term crowd computing.

With the fast emergence of crowd computing, a flood of problematic aspects such as the privacy of the individuals [85], the security of those systems[137], their performance [125] and the quality of their output[71] have arisen. As shown in Figure 1.1 which depicts

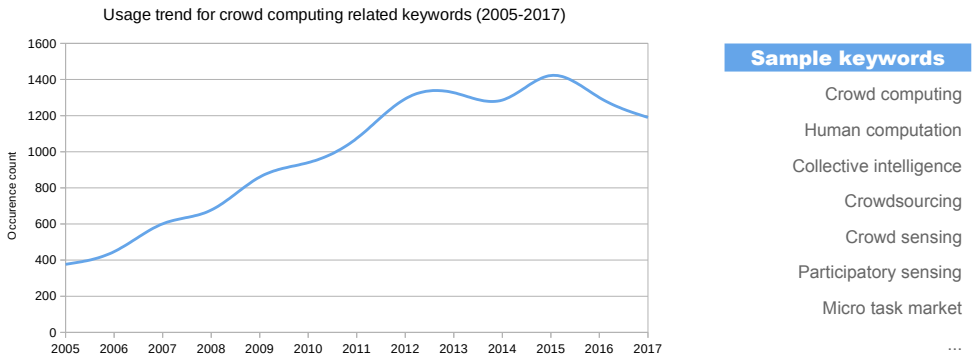


Figure 1.1: Literature usage trend for crowd computing related keywords between 2005 and 2017.

the usage trends of crowd computing related keywords in literature[1][2], these challenges have fostered the research in crowd computing which in turn has led to a rich literature and lit on open challenges to be tackled. However, before dealing with any of these challenges one must draw the limits of his maneuver zone and set some definitions. That is because, despite being around for almost two decades now and in spite of showing different challenges, requirements and characteristics, these systems are still lacking for a clear delimitation unanimously adopted by the research community. In this work, we consider a specific variant of crowd computing which is *crowdsourcing*.

1.1.2 Crowdsourcing: a definition

Jeff Howe [46] - who first introduced the term "Crowdsourcing" in 2006 - defined it as:

> "*the act of taking a task traditionally performed by a designated agent (such as an employee or a contractor) and outsourcing it by making an open call to an undefined but large group of people...*"

Twelve years later, and from a conceptual perspective, this definition still holds the essence of what crowdsourcing is i.e., outsourcing tasks to the crowd. Nevertheless, with the establishment of specific tools i.e., platforms and services to implement this outsourcing approach, it is obvious that this definition became loose from a practical and theoretical perspective. This is clearly reflected in the literature where the term crowdsourcing is mostly used to refer to micro-task markets like Figure Eight and Amazon Mechanical Turk [6, 20, 37, 40, 76, 122, 125], and where other outsourcing-based approaches, formerly perceivable as crowdsourcing, are nowadays referred to with more specific terms such as *Crowd sensing*, *Co-creation* and *open innovation* platforms. Therefore, a refinement of the original definition is needed. First and for the sake of precision, we set the following terminology which will be used in the proposed definition and further in the rest of the thesis:

[1] Results from the ACM Guide to the Literature counted on 06/03/2018. URL : `https://dl.acm.org/advsearch.cfm?coll=DL&dl=ACM`.

[2] The slight decrease in the keyword occurrence might by due to the indexing delays.

Terminology 1.1.1

A requester is a physical person or an organization connected to a crowdsourcing platform. They design and submit the tasks and pay the workers through the platform.

A worker is a physical person connected to a crowdsourcing platform. He perform the tasks.

A crowd is the set of workers of a given crowdsourcing platform.

A contribution is an answer given by a worker to a given task.

Second, we define crowdsourcing as follows :

Definition 1.1.1 **Crowdsourcing** is a technique that consists in outsourcing a task by making an open call to an undefined but large group of incentivized workers. The *Open call* is web-enabled, explicit and limited in time, the *tasks* are human intelligence tasks, well defined, concise and device independent and the workers are incentivized by a monetary *reward* received upon completing the task.

According to this definition, we clearly differentiate crowdsourcing from the following related domains: (i) In **crowd sensing** (a.k.a participatory sensing), a worker is only responsible of taking the decision of contributing or not. The actual contribution is sourced from her device sensors. Hence, the task is not device independent and the workers are not the source of the collected knowledge. (ii) In **co-creation platforms**, people are explicitly asked to contribute [120, 134]. However, the call is permanent and the contribution task is not precise e.g., in Wikipedia, contributors are asked to write and edit articles; neither topics nor deadlines are specified. Furthermore, contributors in these platforms are usually incentivized by their interest in the community and topic instead of monetary rewards. (iii) **Open-innovation platforms** and **macrotask crowdsourcing** are aimed at finding solutions for complex design and knowledge-heavy problems [41, 51]. In contrast with crowdsourcing, open-innovation tasks are not concise and in some cases[3], the reward is perceived only by those who find the best solution. (iv) In **stealth crowdsourcing** or **side effect computing** [21, 126], tasks are hidden in various purpose application e.g., a game like the ESP[4] game [126] or a security measure like ReCaptcha[5] [127]. Hence, workers are not incentivized by their quest for monetary reward but by their willingness to play or to complete the login step instead.

1.1.3 Crowdsourcing in use

Figure 1.2 depicts a generic workflow of a crowdsourcing process (Steps 1 to 7). Typically, all the interactions between the requesters and the crowd occur over and are controlled by a crowdsourcing platform. The requester who holds a complex or large scale problem to

[3] e.g., innocentive (`https://innocentive.com`).

[4] An image content guessing game. Answers proposed by the players were later used to label a large image dataset.

[5] A verification process used to avoid bot access to online content that helped digitizing old printed material by asking users to decipher scanned words from books that computerized optical character recognition failed to recognize.

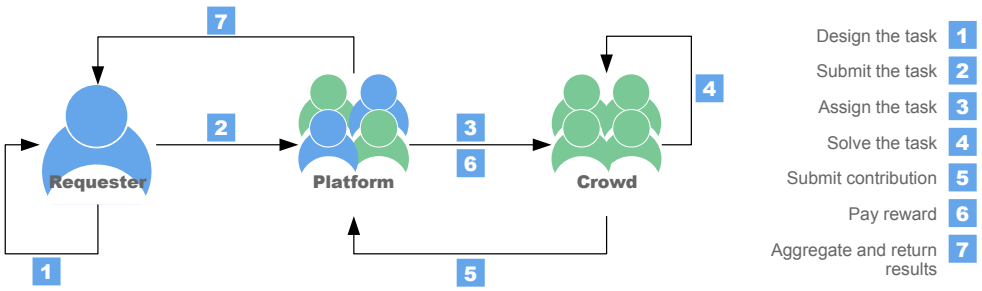


Figure 1.2: A typical workflow of a corwdsourcing campaign.

resolve, starts by dividing this problem into smaller, standalone and brief tasks (Step 1). He submits those tasks to the platform and configures the campaign to fit his needs in terms of quality and budget e.g., number of contributions per task, reward per worker per task, etc. (Step 2). The tasks are then published to the crowd and iteratively[6] assigned to the workers (Step 3). The task assignment can be *straightforward* i.e. it depends only on the worker decision of contributing, or *conditioned* i.e. it depends on the requester's configurations and worker's qualification. The workers submit their contributions (Steps 4 and 5) and receive their rewards (Step 6). The assignment process continues until all the tasks receive the needed number of contributions. Contributions from different workers are then aggregated to infer one solution for each task and a final report with the individual and the aggregated results is submitted to the requester (Step 7).

The efficiency of crowdsourcing as well as the emergence of easy to use platforms have democratized its usage. Applications nowadays broadly covers data labeling, data validation, multimedia transcription, artifact creation, corpus translation as well as user surveys. This wide spread usage is reflected by the fast growth of its market which achieved, in the US only, an annual growth of 37.5% over the last 5 years (2012 - 2017) and reached a revenue of USD 1 Billion. For reference[7], this compares to the USD 411.5 Million blockchain (global) market, to the USD 1.47 Billion (global) edge computing market, and to the (estimated, global) USD 471.2 Million deep learning market [36, 49, 78, 116].

Labeling large dataset: a use case

In an era where Artificial Intelligence is emerging at a steady and fast pace through its underlying concepts such as machine learning and data mining, and where the semantic web has taken over the web technology landscape, the quest for collecting labeled data is a persistent and fundamental task for researchers in these domains. In the last decade, crowdsourcing has proved its ability to address this challenge by providing a means to collect labeled data of various types, at a low cost and short time as compared to expert labeling [47, 94].

[6] For instance, to deal with people leaving and not finishing.

[7] Picked among the top trending technologies based on the Gartner Hype Cycle `https://www.gartner.com/smarterwithgartner/top-trends-in-the-gartner-hype-cycle-for-emerging-technologies-2017/`.

1.2 THE QUALITY ISSUE IN CROWDSOURCING SYSTEMS

Despite being structurally simple, crowdsourcing systems face complex issues because they involve two important sources of uncertainty and ambiguity i.e. the tasks and the workers. On the one hand, tasks might be unclear, very generic or specific, badly written, poorly designed, intrinsically difficult or subjective. Workers, on the other hand, might be more or less qualified due to their various educational and professional qualifications, interests or former experience in crowdsourcing. They can also be biased with respect to the task content and type because of their preferences and demographics e.g. age, gender, country, etc. Moreover the crowd i.e. the group of workers, when regarded as a whole, is characterized its complexity. In fact, the accuracy with which a given task is achieved depends not only on the individual contributions of the workers but on their distribution in the crowd and on the manner the task is presented to - and selected by - them at the moment it is submitted as well [1, 3, 20, 68]. All these factors generate a noisy and low quality crowdsourcing output which in turn impacts its usability potential. For instance, when it comes to labeling a training dataset for a given machine learning approach, a poorly labeled dataset could lead to the generation of bad, thus unusable, models. Hence, the quality of the data produced through crowdsourcing is still questionable especially when the task is subjective or ambiguous or requires a minimum level of domain expertise [111]. Therefore, there is a need for verifying the quality of crowdsourced data. We define the quality control in crowdsourcing as follow:

Definition 1.2.1 In crowdsourcing, **quality control** refers to mechanisms employed by the platform to ensure that at the end of a campaign, an optimal number of correct answers can be guessed for the submitted tasks. We exclude from this security mechanisms used to identify security threads such as malicious contributors, Sybil attacks, collusion attacks, etc.

Indeed, this quality control is subject to practical constraints which we analyze below:

Cost : The budget payed to complete a crowdsourcing campaign can be divided into two parts: (a) the *contribution fees* which consist of the sum of rewards payed by the requester to the contributors for finishing the task and (b) *service fees* which are a fixed rate[8] charged by the service provider i.e. the platform. In practice, the *contribution fees* include, for quality control purposes, an implicit cost e.g. multiple contributions per task, an explicit cost e.g., only accepting high level workers which are better payed than an average worker, or both of them. In some cases like the labeling of large dataset use case, these implicit and explicit costs can rapidly grow to reach fairly high sums. For instance, asking for 4 contributions per task instead of 3 adds a 33% of additional cost to the overall campaign cost. If we consider a large dataset of 1 million entries, it is obvious that this additional cost can be high. One major advantage of crowdsourcing is its cost-effectiveness. Therefore, it is crucial that the budget proportion related to the quality control does not exceed a certain threshold after which the quality to cost ratio of crowdsourcing is no more advantageous compared to an expert labeler (alone or assisted by an algorithm).

[8] e.g., Figure Eight charges for the service a net amount equal to 20% of the global sum of rewards.

Time : The time needed to control the quality of contributions is crucial in some cases such as rescue and relief related crowdsourcing (e.g. collecting ground information, detecting interest points in images, ...). However, it is still important especially in the case of large campaigns like the labeling of large dataset use case. A common practice when implementing this kind of campaigns is to launch the tasks by batches or to a batch of workers at once. This allows for a better design related trial and error approach. Since crowdsourcing platforms are very dynamic in term of task and worker arrival and departure[20], workers and workers' availability change a lot. Therefore, controlling the quality every time a batch is launched is mandatory. If we consider that a quality-control-related time overhead exists, it is clear that the overall time overhead of this sequential workflow can heavily impact the campaign completion time.

A priori knowledge : Historical knowledge about user preferences and performances is a common and efficient aspect leveraged by recommendation[7], security e.g. login history, and quality [25] methods in user-centric and user-enabled systems. However, these methods usually suffer from a cold start problem. In crowdsourcing, this cold start problem raises from the fact that we ignore the worker global performance if he has few and/or sparse historical contributions. Consequently, controlling the quality should not totally rely on this kind of knowledge.

1.3 Dealing with the quality issue in crowdsourcing systems

Many approaches have been proposed in the literature to deal with the quality issue in crowdsourcing. We divide these approaches into two categories: Aggregation optimization and Worker selection.

Approaches in the first category focus on optimizing the contribution aggregation process. Early works used majority voting (MV) with multiple assignments to infer the correct answer to a given task. However, this technique considers all workers, thus their contributions, to be equally reliable, which is not realistic. Giving different weights to the different votes remedies to this limitation and improves the quality of the aggregation. In [56, 79, 108], authors leverage different "accuracy features" such as graded and binary accuracy, mobility patterns and in-user-interface worker behavior (e.g. number of clicks and mouse movements) to explicitly weight the contributions of each worker. More widely used techniques [6, 76, 131] rely on probabilistic data completion methods like the expectation maximization algorithm (EM) [14, 15] to implicitly weigh the contributions and infer the correct answers. In these cases, the weights and the correct answers are simultaneously inferred by maximizing a likelihood model. The accuracy of the inference process depends on two factors; the campaign elements to be modeled (the worker, the task or both) and the parameters used to model them. Some methods propose to add more knowledge to this process using multiple stage crowdsourcing such as the produce/review workflow described in [6]. Adding a review stage is another way to increase the confidence in the aggregation output, however, it increases the time (since it is a sequential process) and the budget (since the reviewers are paid) needed to complete the crowdsourcing campaign.

Worker selection approaches aim to find in the crowd the set of the most reliable workers to solve the task. To this end, early commercial crowdsourcing platforms used worker

screening, i.e., selecting workers based on qualification tests and filters . Screening is usually done before the worker is allowed to solve the actual tasks. Once the worker passes the screening, his contributions are no more controlled and hence, he has no obligation to submit accurate responses. Li et al. [76] select reliable workers based on their declarative profile (e.g., age, gender, education, ...) and their performance in a probing stage. By targeting a specific group, one can reduce the number of assignments and, as a result, decrease the budget. However, because of the probing phase, the budget still remains high and the task completion time is increased. Roy et al. [107] select workers whose skills matches the skills required for a given task. These methods consider that the knowledge about the worker skills exists, which might not be always true and count on manual tagging to determine the skills needed to finish a task, which might not be precise.

1.4 Research challenges and thesis contributions

In this thesis, the research problem we tackle can be summarized with the following statement:

> How to control the quality of contributions in a crowdsourcing campaign, while minimizing the budget and time needed to complete it and being agnostic to historical knowledge about the individual workers?

Indeed, optimizing the aggregation process is an effective way for increasing the quality of the final output. However, since the aggregation step is performed upon task completion, these methods are not able to control the access to the tasks and thus to restrict the budget and time needed to complete the crowdsourcing process. By contrast, worker selection approaches control this access and thus allow to better master the budget and the time. Nevertheless, existing worker selection approaches are not optimal (in terms of time and budget needed to complete the task) because they deal with the particularity of each task in an extreme manner. Some ignore it and consider that all tasks are similar from a same worker perspective - that is, a worker who proved to be good in the tasks to which he already participated is certainly good for any upcoming task whatever its type is - which reduces the performance of quality control. Others consider each task to be unique, i.e., the performance of a worker needs to be reevaluated for each incoming task, which increases the budget and the time of completion. However, in practice tasks are not all similar, yet sometimes they share "similar traits". Furthermore, a worker usually shows a stable performance in completing similar tasks. Taking advantage of these properties can help bypassing this extremeness in dealing with tasks. Thus the first question to be answered in order to solve our research problem is the following one:

> **RQ. 1** Is it possible, in crowdsourcing, to leverage the similarity between historical tasks and incoming tasks to improve the online worker selection process and to reduce the time and the cost of crowdsourcing campaigns?

When evoking the similarity of the tasks, an intuitive question regarding the features based on which this similarity is computed rises. We formulate it as follows:

RQ. 2 Being textual documents, are the existing text vectorizing method suitable for characterizing and comparing crowdsourcing tasks? If not, what is a suitable feature set to represent them?

The second question that should be answered to solve the research problem concerns the dependency on knowledge about workers. Indeed, in order to select reliable workers, a set of performance distinguishing features must be used. As explained in Section 1.2, knowing the historical performance of a given worker can help predicting her future performance. Nevertheless, this knowledge is usually either insufficient or completely absent (for new workers). Therefore a set of history-independent features must be exploited during the selection process. Self-declared features such as skills and demographics are easy to collect and history independent features. Moreover, these features can be directly related to the worker performance in certain tasks. In this context we formulate our third research question as follows:

RQ. 3 How to select high quality workers in crowdsourcing based on self declared features?

In response to these questions, this thesis brings two contributions:

C. 1 Context Aware Worker Selection (CAWS): An offline-learning/online-selection framework for a cost and time efficient quality control.
In response to RQ. 1, 2 and 3

We propose a framework to learn during an *offline* stage the relation between the various types of historical tasks and the declarative profiles of reliable workers. The task types are determined through a content-based clustering process. The learned models are then used in an *online* stage to select reliable workers within the available crowd. First, by taking the learning process offline, the time of task completion is reduced (as there is no need for an online probing step). Second, by using historical tasks to perform the learning process, incoming tasks are only crowdsourced to the needed extent and thus the cost of task completion is reduced. Finally, using the declarative profiles renders the learning and selection process agnostic to any knowledge about the previous performance of workers.

It is clear that our method exploits a wide scope of aspects found in a crowdsourcing system, i.e. the task content, the worker profiles and the worker reliability. Thus, its evaluation requires a dataset that contains, quantitatively and qualitatively, all of these information. This raises the need for a rich evaluation dataset. Existing datasets [16, 53, 57, 68, 76, 117, 129] are not satisfactory quantitatively and qualitatively to evaluate our workflow and to compare it to state-of-the-art quality control approaches which motivates the following contribution:

C. 2 CrowdED: A rich dataset to evaluate and compare our worker selection approach.

CrowdED fulfills three comprehensive specifications that we summarize as follows: (i) **Data richness** meaning that CrowdED is rich in information about the workers' profiles and

the tasks' content. (ii) **Data diversity** meaning that CrowdED reflects the diversity of tasks and worker profiles in a real crowdsourcing system. (ii) **Contribution abundance** meaning that CrowdED provides a large number of contributions for each of its worker and tasks.

We complement our work by a software contribution consisting of an open source platform that allows a budget and type constraint creation, extension and enrichment of crowdsourcing quality control evaluation datasets similar to CrowdED:

C. 3 CREX: A platform to collaboratively extend and enrich CrowdED.

Although the literature is rich in quality control approaches, there is an important gap in the crowdsourcing research community preventing a sound evaluation and comparison of those methods, that is the lack of sufficient evaluation datasets. In fact, some of those methods have been evaluated on quantitatively and qualitatively limited (i.e. no information about workers, tasks or both of them) datasets and others have borrowed datasets from other related domains such as machine learning and recommender systems. While in the former case, there is a lack of information needed by our framework, the representativeness of the datasets can be questioned in the latter. Moreover, the literature does not provide any tool to extend and enrich these datasets (to be exploitable by other methods). The added value of CrowdED comes from the fact that besides helping in evaluating our approach, it is rich and big enough to be used in evaluating and comparing a large number of existing crowdsourcing quality control methods. Besides, CREX helps a collaborative extension and enrichment of CrowdED which ensures it is a future-proof dataset. All of this helps bridging the dataset related gap mentioned earlier.

1.5 Thesis outline

This chapter presented the context of this thesis by describing crowdsourcing systems, the quality problem they suffer from and the constraints shaping the quest for an efficient solution of this problem. In this thesis, we present our contributions toward finding this solution. The following content is structured as follows:

- Chapter 2 discusses the literature of quality control in crowdsourcing and shows the limitations of the state of the art methods. Moreover, it introduces and motivates the positioning of this work within the quality control literature.
- Chapter 3 introduces our Context Aware Worker Selection method (CAWS) and details its learning and targeting phases.
- Chapter 4 presents the challenge of creating a rich and extensible evaluation dataset and introduces both, the Crowdsourcing Evaluation Dataset (CrowdED) and the CReate Enrich eXtend platform (CREX) along with their theoretical and technical detail.
- Chapter 5 details the experimental setup and the evaluation process of CAWS as well as the results of these evaluations.
- Chapter 6 concludes this thesis and provides an insight into the potential improvements of this work and the perspectives it opens.

List of publications related to this thesis

- Tarek Awwad, Nadia Bennani, Konstantin Ziegler, Veronika Rehn-Sonigo, Lionel Brunie and Harald Kosch. **Efficient worker selection through history based learning**. The 41st IEEE international Conference on Computers, Software and Applications (Compsac 2017).
- Tarek Awwad, Nadia Bennani, Lionel Brunie, David Coquil, Harald Kosch and Veronika Rehn-Sonigo. **Task Characterization For An Effective Worker Targeting In Crowdsourcing**. The 17th IEEE High Assurance Systems Engineering Symposium (HASE 2016).

List of resources related to this thesis

- CrowdED on Github: `https://github.com/Project-Crowd/CrowdED`
- CREX on Github: `https://github.com/Project-Crowd/CREX`
- CAWS on Github: `https://github.com/Project-Crowd/CAWS`
- Project-Crowd.eu: `https://Project-Crowd.eu`; This site groups all the material related to this thesis such as reports, slides, demos and tutorials of CREX-D and CREX-C as well as a demo of the crowdsourcing campaign.

2

State of the art

The usefulness of crowdsourcing heavily depends on two aspects : first, the quality of the produced output and on how it compares to an expert contribution and second, the cost and the time needed to collect this output. This makes it fundamentally important to control the quality of the worker contributions while keeping the time and cost overheads low. In Chapter 1 of this thesis, we briefly cited the existing quality control approaches in crowdsourcing and highlighted their limitations in terms of time- and budget-overhead optimization. In this chapter, we further review these approaches and describe in detail the state of the art methods of quality control. To this end, we propose a four category classification of those methods: *Aggregation optimization techniques, worker selection approaches, individual performance enhancement, and multi-stage crowdsourcing*. We draw a particular focus on worker selection approaches as they directly relate to this work. The limitations of these methods in terms of quality, time and cost efficiency are also discussed.

The primary contributions of this chapter can be summarized as follows:

1. We survey and detail quality control methods in crowdsourcing and discuss their contributions in terms of output quality.

2. We propose a classification of these methods from the perspective of their occurrence time on the crowdsourcing timeline and discuss the impact of this occurrence time on the time and cost overhead they add.

Roadmap. Methods for aggregation optimization, worker selection and screening, individual performance enhancement and multi-stage crowdsourcing are described and discussed in Section 2.1, Section 2.2, Section 2.3 and Section 2.4 respectively. Then, in Section 2.5, the positioning of this thesis is advocated based on a discussion about the impact of the method occurrence time on time and budget overheads and on the advantage of worker selection on the accuracy of the aggregation algorithm. The chapter is then concluded in Section 2.6.

2.1 Aggregation optimization approaches

In practice, the ultimate goal of a crowdsourcing campaign is to find the correct answer for each task in this campaign. Indeed, one cannot rely on the answer of one or few workers since this is prone to error. That is because workers in crowdsourcing are less likely to be experts. Therefore, crowdsourcing relies on collecting redundant contributions for each task. Those contributions are then aggregated which yields the correct answer for the task. Practically, the aggregation problem is the process of inferring the true labels from noisy observations which are the worker contributions [121]. A fair amount of methods in crowdsourcing quality control has focused on optimizing the aggregation step in order to improve the output of the crowdsourcing process [6, 56, 58, 62, 73, 76, 100, 121, 131]. In general, the proposed aggregation methods can be classified into two categories: Iterative and Non-Iterative [48].

2.1.1 Non-iterative aggregation

As defined by Hung et al. [48], non-iterative aggregation methods *"use heuristics to compute a single aggregated value of each question separately"*. It consists in finding, in a single iteration, the answers for each task out of the provided contributions. One common non-iterative aggregation technique is Majority Voting (MV). Given a task and a set of answers for that task, MV aggregation consists in selecting the most voted answer in the worker contributions. In classical MV, all the contributions are considered to be equally reliable. That is, the reliability of the worker who provided a contribution is not regarded when counting the vote. This renders the majority voting vulnerable to malicious votes and prone to high rate of errors for difficult tasks. To cope with these limitations, one can give higher weights for more reliable answers when computing the final vote. This is known as Weighted Majority Voting (WMV) and has been repeatedly used in the literature [44, 56, 74]. MV and WMV can be formally expressed as follows :

Let t be a task.
Let W be the set of workers who answered the task.
Let A be the set of unique answers given by the workers for t.
Let c_{wt} be the contribution of the worker w for the task t.
Let v_w be the weight of the worker w.

The estimated correct answer of t, denoted $\hat{a}_t$, yielded by the **MV** is given by:

$$\hat{a}_t = \arg\max_{k \in \mathbb{A}} \sum_{w \in W} I(c_{wt}, k) \tag{2.1}$$

and by the **WMV** is given by:

$$\hat{a}_t = \arg\max_{k \in \mathbb{A}} \sum_{w \in W} v_w \times I(c_{wt}, k) \tag{2.2}$$

where I is an indicator function such that:

$$I(x,y) = \begin{cases} 0 & \text{if} \quad x = y \\ 1 & \text{otherwise} \end{cases} \tag{2.3}$$

Various measures can be used as weights for the contributions. For instance, Figure Eight uses a "contributor trust" computed as the rate of correctly answered test questions [24]. This test question principal is also called "Honeypot" [72] or gold-based quality assurance [68]. Practically, this consists in continuously measuring the accuracy of the worker using test tasks - with known answers - randomly injected in the task completion process. The measured accuracy is then used as weight to all of the contributions given by the worker. In [62], Khattak et al. use the test questions not only to estimate worker accuracy but to also to compute an estimate of the difficulty of the remaining (unlabeled) tasks. This can be summarized as follows: the workers' accuracy for the test questions is computed. Then temporary labels for the unlabeled tasks (i.e., all the tasks except the test questions) are computed through the classical weighted majority voting detailed earlier in this paragraph. Those temporary labels are then used to estimate a difficulty score for each task by computing the ratio of workers who correctly answered it (w.r.t. its temporary label). Finally, another weighted majority vote occurs to find the final estimated answers. In this final vote, both the worker accuracy and the task difficulty are used as weights for the contributions. In a more recent work [63], Khattak et al. use the same aggregation technique, yet, an entropy measure is used to cope with the uncertainty of estimating worker accuracy and task difficulty from a small sample.

2.1.2 Iterative aggregation

In contrast to non-iterative approaches, iterative aggregation is based on multiple computational rounds. In each of these rounds, the probability of a set of options being the set of correct answers is updated. These probabilities are incrementally updated until convergence of a certain objective function [48]. Generally, in the literature, iterative aggregation methods [76, 98, 99, 100, 131] are based on the Expectation Maximization algorithm (EM) [14, 15, 83]. Broadly speaking, the EM algorithm is a generative model that aims at finding missing data while estimating the hidden parameters of a statistical model from a set of available observations. As its name hints, the EM algorithm performs two steps in each iteration: an *Expectation* step and a *Maximization* step. Figure 2.1 depicts the workflow of the EM algorithm. After an initial estimation of the model parameters[1], the expectation step computes an estimate of the missing data. Then, during the maximization step, the model parameters are updated through a maximum likelihood estimation, based on the data estimated in the expectation step. These two steps are repeated until the algorithm converges (i.e., the parameters and data estimation are unchanged).

Hereafter, we instantiate this workflow on our use case of interest i.e., the contribution aggregation. In fact, the EM algorithm - with its different variations - is widely used and approved in the crowdsourcing community [13, 76, 101, 102, 117, 146, 147]. Here, the missing data are the correct answers of the tasks. The parameters of the model are the

[1] Usually the initialization is done randomly, unless hints about the probability distribution of those parameters exist.

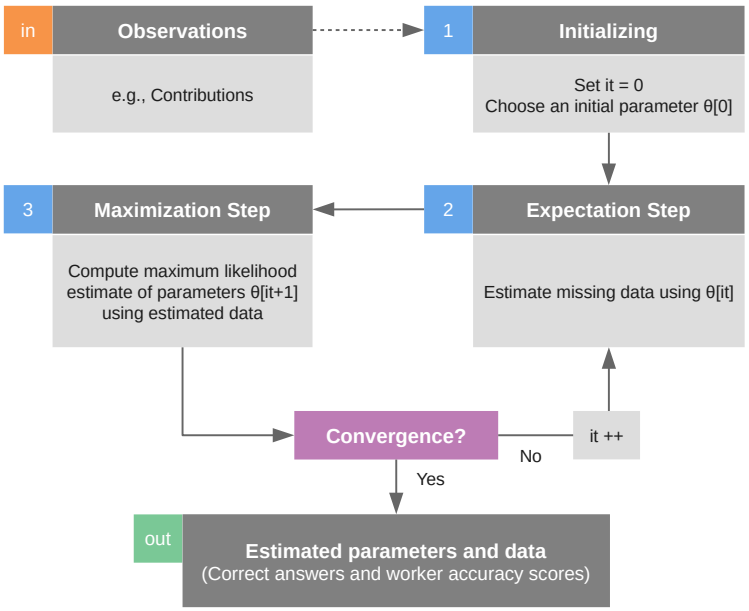


Figure 2.1: The workflow of the EM aggregation where $\theta[it]$ represent the model parameters to estimate e.g., worker accuracy scores.

workers' reliability parameters while the observations are the contributions themselves. In each iteration, the worker reliability parameters e.g., their accuracies, are used as weights for their contributions in order to compute an estimate of the correct answers. This estimate is then used to update the worker reliability parameter until those, as well as the estimated correct answers, are unchanged.

Using merely the accuracy of the workers as weights for their contributions in the aggregation process is the simplest way of creating the statistical model of an EM based aggregation. However, it is possible to improve the quality of the aggregation result by modeling other parameters of the system as latent variables to be estimated during the aggregation process. For instance, Li et al. [74] used both the accuracy and the inaccuracy of the workers in weighting their correct answers and their bad answers respectively. Raykar et al. [100] proposed a similar algorithm except that it is optimized for binary labeling tasks as the workers are modeled by their *sensitivity* i.e., the ratio of positive answers which are correctly assigned and their *specificity*[2] i.e., the ratio of negative answers which are correctly assigned. Whitehill et al. [131] proposed a more complicated model where, in addition to the worker accuracies, the difficulty of the tasks is used as weight for the contributions. This model has been extended by Jin et al. [53] who proposed to leverage side information about worker and tasks to better encode the worker bias and expertise as well as the item difficulty used in the EM model. In [98], the authors proposed a version of EM that handles the ordinal labels instead of the categorical labels supported by the classical EM algorithm.

In fact, there is no theoretical guarantee of an optimal convergence of EM. The algorithm can get trapped in a local optimum which yields a suboptimal estimation of the correct

[2]The sensitivity and the specificity are also know as true positive rate and true negative rate respectively.

answers [59]. This is specifically true for non-binary labeling tasks [144]. Just a few works in the literature tried to tackle this problem in the context of crowdsourcing. Ghosh et al. [39] presented a spectral algorithm that provably learns the true item qualities with bounded errors. Yet, this assumes that each user performs a large number of tasks. Zhang et al. [144] used a spectral method to improve the initialization of the worker accuracies which improved the convergence rate of the EM algorithms.

2.2 Worker selection approaches

Aggregation optimization methods aim to control the output quality of the crowdsourcing by maximizing the information gain in the already collected contributions. Meanwhile, worker selection approaches try to improve the quality of the individual collected contributions. Roughly speaking, those approaches consist in allowing only reliable workers to participate in the task solving process.

2.2.1 Worker screening

The most basic means of selecting worker is to deny the access to a task for unqualified workers, i.e., screening them. Worker screening has been widely used in commercial crowdsourcing platforms, each of which has defined the qualification differently. Figure Eight [32], Amazon Mechanical Turk (AMT) [123] and FreeLancer [35], for instance, all use the Overall Approval Rate (OAR). OAR is an individual trustworthiness measure computed using the historical participations of the worker in solving tasks on the platform. Generally, OAR is the percentage of the correct answers in all of the submitted answers [138]. It can also be the combination of this percentage with the number of solved tasks as well to reflect the experience of the worker in the task solving. This is done in Figure Eight where workers are mapped, based on their (combined) OAR, to three levels ranging from 1 (lowest reliability) to 3 (highest reliability). Upon task submission, a requester is able to chose the minimum required level of qualified workers [25]. While reliable for eliminating low quality and malicious workers, OAR is not sufficient to determine the real quality of a given worker or to differentiate two workers with fairly similar OAR [138]. Moreover, as it is computed globally in terms of the worker's contributions, this measure is not suitable to differentiate the workers for specific tasks or even to assess the real reliability of the worker who might have participated merely in tasks which he finds easy and to which he is accustomed. Furthermore, OAR is prone to the cold start problem as it is not computable, or at least not significant enough, when it comes to new workers.

Besides the OAR, Figure Eight allows to filter out the workers using basic rules such as country, language and channel [3]. AMT adopts a similar, yet more flexible, filtering system where each worker is characterized by a set of qualifications such as skills, abilities and reputation. Qualification requirements are defined during task submission, where a requester can require a given qualification, such as the language, to be present in the worker profile before allowing him to see and enter the task. The difference with the worker filtering settings of Figure Eight is that the requesters can define their own custom qualification type

[3] Bigger crowdsourcing platforms rely on smaller crowdsourcing markets called channels. These channels act as a broker allowing an easier and locally adapted (e.g. by country) crowd recruitment.

[22]. When compared to the OAR, worker filtering can better fit, without being optimal, the specific requirements of each task. For example, let us consider a decision making task related to American football where the workers are supposed to analyze an in-game situation and provide the decision they would take to succeed the action such as punt or pass. In this case, workers situated in countries where American football is a common sport are more likely to have better knowledge in this domain and thus, being more reliable for completing the task. Nevertheless, this filtering is manual and depends on the requester's subjective judgment. This makes it a time consuming and possibly biased process.

In order to consider the specificity of each task set during the worker selection, quiz tests might be also used. They consist in testing the reliability of the workers before they enter the task set using a small amount of tasks sampled from the set. The correct answers for those test tasks can be collected through expert labeling. Workers who fail Quiz Mode are not paid and are not allowed to work on the tasks. This technique is also used by Figure Eight [27] and by AMT as part of their qualification system. Quiz tests require labeled tasks which might be costly to acquire; in practice, when expert labels are not available, it is common to crowdsource the quiz tasks by a large number of workers (up to 500) in order to collect their true labels. Moreover, the choice of test tasks might be biased and thus, those do not reflect the real difficulty or specificity of the task.

2.2.2 Worker tracking

In practice, worker screening is done before the worker is allowed to solve the actual tasks. Once the worker passes the screening, his contributions are no more controlled and hence, there are no obligations to submit accurate responses. To tackle this limitation, one can use gold-based quality assurance [68], also known as seeding [60]. It consists in continuously measuring the accuracy of the worker using test tasks - with known answers - randomly injected in the task solving workflow. A high error rate causes the rejection of the worker from the current campaign. Similar to the quiz tests described earlier, gold-based assurance depends on the quality of the task questions. On the Figure Eight platform, most jobs have between 50 and 100 test questions [26]. This implies an additional work load on the requester side and makes the selection either prone to requester bias in the provided gold answers or expensive if those gold answers are collected through crowdsourcing with high redundancy. Moreover, while guaranteeing that the worker performance will not maliciously drop after he has been selected for that task like it might be the case in worker screening, test questions in this technique are distributed over a long period within the job solving process. For instance, by default, Figure Eight uses one test question per task page [4], which means that some workers might spend a fair amount of time solving the tasks before getting eliminated from the job because of a quality drop. On one hand, this increases the time of job completion as eliminated workers must be replaced and, on the other hand, this is unfair regarding the time the worker spent doing the job without being paid.

In [108], Rzeszotarski et al. argue that the reliability of a worker can be measured through a set of behavioral features related to his interaction with the user interface during the task solving. To accomplish this measurement, the worker behavior e.g., time for completing a task, number of clicks, mouse travel, etc. is tracked in the platform interface using software

[4] A page consists of 1 to 19 tasks [26].

plug-ins. The collected features are then aggregated which results in a reliability score of the worker. This method, called fingerprinting, can eliminate spammy contributions because some malicious behavior in the context of crowdsourcing are easy to define and filter e.g. filling the answer randomly, finish the task in record time, use key strokes such as *TAB+SPACE* combinations etc. However, it is fairly difficult to model the behavior of trustworthy (non-malicious) workers, using these features in a way to reflect their real performance.

In [132], Whiting et al. introduce a reputation system based on guilds. Their approach consists in splitting the workers into groups, each of which tries to manage and improve the expertise of its members. These groups certify their members by internally rating the contributions of those members for peer assigned tasks. These ratings are then translated to guild levels which in turn serve as reputation signals on the crowdsourcing platform. It is worth noting that worker screening and worker tracking systems are also considered as reputation systems as they are used in the literature to compute a measurable trustworthiness of crowdsourcing workers [55, 81, 118].

2.2.3 Skill matching

As stated earlier, worker skills can be used to filter out reliable workers for a given task e.g., in AMT's qualification requirements. This filtering process consists of deciding whether a worker should be allowed or denied the access to a very specific task. Nevertheless, one can see the system in its big picture and try to optimize the task assignment process by matching the available workers to the available tasks. Skills have been leveraged in a number of approaches in order to perform this task-to-worker matching [80, 96, 107, 113].

In [107], Roy et al. modeled the task assignment problem as an optimization problem that takes into account the worker skills, their wage requirements, and their availability. This work can be summarized as follows: for a given workload i.e., a number of active tasks, i.e., unfinished tasks, in a given period, the framework called *SmartCrowd* tries to build an index of the tasks in a way to account for, on the one hand, the minimum skills required for the task and the maximum possible pay and, on the other hand, the skills of the available workers and their requirements in terms of wage. The index of each task is a weighted linear combination of skills (the aggregate skills of all selected workers) and wage (the aggregate wage of all selected workers). The objective of the optimization process is to maximize the sum of the indices of the tasks in the workload. The authors proposed three optimal and approximative algorithms to solve this problem. The approach is tailored for knowledge intensive crowdsourcing where workers build on each others contributions to gradually increase the quality of each task answer (thus, aggregating the skills of the selected workers is possible). However, one cannot apply this approach on more classical, and more common, tasks such as the use case of labeling large datasets for which the worker contributions are independent and collaboration is absent.

Rahman et al. [96] followed a similar optimization approach, also applied on crowdsourcing tasks where collaboration between workers is needed. However, in order to take the collaborative aspect into account, they also accounted for the affinity between the selected workers. The pairwise affinity between workers is computed as their similarity using simple socio-demographic attributes, such as region, age and gender. This allows also to define an

intra-group affinity measure i.e., the diameter of the group of worker selected for a given task and the inter-group affinity between two group of workers i.e., the sum of pairwise affinities of workers, working on two parts of the same task [5].

In [80], Mavridis et al. proposed to use skill taxonomies in order to characterize the workers and the tasks. Then, based on this taxonomy, they propose to perform a one to one matching between the worker set and the task set. This is done using one of three algorithms that consist of two heuristics and one optimization algorithm. First, a heuristic called *MatchParticipantFirst* which tries to assign the most specialized tasks first, to the participants with the lowest number of skills (this saves most diverse participants for other tasks). Second, a heuristic called *ProfileHash* which tries to achieve perfect matches and gradually loosening the constrained by considering generic-skill based matching. And last the Hungarian match, which is a combinatorial optimization algorithm often used to find perfect matchings with minimum normalized cumulative distance. However this work assumes that a task can be characterized by a single skill which is not realistic (as a task can be more complex and links to many skills) and that a task is mapped to only one worker which is not true in practice. Similarly, considering that the knowledge about the worker skills is present in the system is not realistic.

2.2.4 Profile based selection

As mentioned earlier in this thesis, usually tasks in crowdsourcing require a human intervention as they are not solvable by algorithms. That is because the answers for these tasks depend on the cognitive skills of the human workers such as decision making, appreciation and so on. These skills are usually built and grown during the human life throughout his experiences in the society, workplace, education etc. Therefore, one can easily imagine that these experiences, which can be reflected in the declarative profile of a worker, influence his cognitive skills and, consequently, his performance in solving the crowdsourcing tasks. This has been studied in [61] where the authors showed that a correlation between the performance of the workers and some of their demographical and personality-related features exists. The main findings of the work is that in terms of demographics, location has a very strong relation to accuracy, with Asian workers demonstrating significantly poorer performance than American or European workers. Additionally, in terms of personality types, they found that openness and conscientiousness relate significantly to accuracy. The authors summarized their overall conclusion as follows : *"there is a complex interaction between the particular HIT [tasks] conditions and the types of workers who engage in a task, and that these worker characteristics are related to the quality of their work"*.

In [76], the authors leveraged these worker features such as age, gender, skills, location, study domain and level etc., to perform a cold-start-free and dynamic worker selection. Their method consists of three stages: the first stage consists of an information gathering phase called **probing stage**. During this phase, a part of the task is distributed to the connected crowd in order to get an unbiased sample of workers. At the same time, the features of each worker are extracted. This is done either by using the data provided in the platform profile or by simply sending a questionnaire to the workers in concern asking for the above mentioned information. The gathered data, i.e., the full sample of the crowd and the set of worker

[5] To allow a better merging of the contributions.

features are used as an input to the second stage i.e., **the discovery stage**. The discovery stage aims at modeling the relationship between the quality of a worker contribution for a given task and the features of this worker. The main contributions of the work reside in this phase during which the quality of each worker's contribution is estimated and the profile-features of interest are determined. As a result of the discovery stage, the identified reliable worker group is then called for completing the task or to start a new task. This call occurs during the so called **targeting stage**. By targeting a specific group, one reduces the number of assignments and, as a result, decreases the budget. However because of the probing phase, the budget reduction is not optimal and the task completion time is increased. Moreover, the accuracy measured proposed in this work is optimized for EM which is not usable on all sorts of output (e.g. unbalanced MCQ).

2.2.5 Preference based assignment

Recommender systems are systems that seek to match a set of items in a system to a set of users in a preference driven manner [7, 12, 112]. In crowdsourcing, recommender methods have been used to assign tasks to workers (or select workers for tasks) [2, 37, 109, 140] while maximizing the satisfaction of the workers from a task preference perspective. Recommender methods assume that workers have a better performance in completing tasks they like. Thus, preference driven recommendation can help improving the quality of the crowdsourcing output.

Ambati et al. [2] proposed, for instance, two recommendation methods which use implicitly computed and explicitly gathered features. The first finds the matches between tasks and workers using a bag-of-words approach. The second uses a trained classifier to decide whether the worker would be interested in a given task or not. Both approaches heavily rely on the history of the individual workers which causes a serious cold-start problem. In [140], Yuen et al. described a probabilistic matrix factorization method to recommend tasks to workers called TaskRec. Their approach uses the implicit worker's ratings inferred from his interaction behavior with the tasks as well as a manual task categorization to model the worker preferences. TaskRec improves - compared to traditional recommender techniques - the recommendations in a dynamic context such as crowdsourcing.

Difallah et al. [18] proposed a method that collect the worker preferences from his social media profiles. These preferences, to which preferences from the historical contributions of the worker are added, are used to build a preference profile. A similarity measure uses the latter to find suitable tasks (i.e., tasks which description are close to the worker profile) which in turn are returned to the worker in a push methodology (i.e., sending a notification about the task submission only to the selected workers). As the authors pointed out in their article, this push strategy may lead to delays in the completion of the tasks. Moreover, there might be a privacy related aspect preventing the workers from giving the crowdsourcing platform the permission to extract preference data from their social media profiles. This is not an issue in the case of declarative profile as workers are aware of what data they are sharing with the platform. Instead of proposing tasks separately, Amer-Yahia et al. [3] described a method that proposes to the workers a summary of tasks which match their skills, preferences and the wage they would accept. The proposed summaries are valid i.e., they contain the number of tasks requested for a work session, representative i.e., they cover the range of tasks available,

and personalized i.e., they contain tasks that match the worker interests. In [1], authors extend this work by studying the effect of task diversity in the proposed summary on task throughput, worker retention, and contribution quality.

2.3 Individual performance enhancement

While some methods optimize aggregation techniques and other improve the quality of the contributions through worker selection, another family of approaches proposes to improve the quality of the individual contributions by enhancing the performance of the workers. This can be done through training, incentives, task design enhancement and priming.

2.3.1 Incentives

In the first chapter of this thesis, we presented a definition of crowdsourcing that allows to isolate the domain in which we are interested in the work from other similar domains (Definition 1.1.1). A key element of this definition is the existence of a monetary reward. In the context of crowdsourcing, this monetary reward constitutes the incentivizing element pushing the workers toward contributing. In fact, this incentive has been also proven to boost the quality of the contributions. Various payment and incentive schemes have been proposed in the literature to boost the performance [19, 42, 65, 77, 128, 136, 139].

In [77], Mao et al. studied three payment schemes as follow: (i) a per-task payment scheme which is equivalent to the classical scheme used in commercial crowdsourcing platforms. (ii) A per-annotation scheme, where workers are paid per annotation in a multi-label task [6]. And (iii) a per-time scheme, where workers are paid for the time they spend solving the task. These schemes were compared to each others and to a non-payment scheme where workers volunteered to solve the tasks. The authors found that per-task payment results in the fastest task completion, but lowest recall. Meanwhile, the per-time payment scheme, caused workers to work more slowly, but with better results. Moreover they found that workers who were paid on a per-task basis, show a greater quality drop as the task gets more difficult as compared to the other schemes.

Harris et al. [42] looked into another payment scheme which consists of 4 variants of the traditional per task payment: the first one is the classical per task payment, the second consisted of a monetary bonus payment for each correct answer, the third consisted of a monetary penalty for each bad answer and last a combined scheme were bonus and penalty can be paid or deduced. The results reported in this work showed that the classical payment model(without any bonus or penalty) performed worst than the other schemes and that cases where a bonus is paid, performed the best.

Difallah et al. [19] also experimented with the bonus payment and proposed milestone-based bonuses. In contrast with the aforementioned methods, this work aims at reducing the time of task completion. The authors argued that by convincing the worker to work longer on a task batch i.e., solve more tasks from the batch, they can avoid the situation where new workers need to be recruited to complete an almost completed batch[7]. To achieve this, the

[6] Such as giving keywords that describe a picture.

[7] A batch or a job is a set of similar tasks e.g. a large number of images to label. When few tasks remains in the batch, new workers are less interested in entering it because the time to spend on getting accustomed to the tasks would not be profitable.

authors proposed multiple bonus schemes such as: (i) increasing bonus i.e., paying more bonus at the end of the batch than in its beginning (ii) milestone bonus i.e., accumulating bonus and paying it occasionally, (iii) training bonus i.e., paying more at the beginning of the bath than at the end [8]. And (iv) random bonus i.e., paying a bonus drawn once at random in a lottery similar manner. Results shows that workers paid at milestones stayed longer which reduced the overall time of the batch completion. While the time is decreased, authors showed that the average worker accuracy does not necessarily improve as a result of the worker staying longer. Yet, the variance of their accuracy dropped for workers who stay longer. This can be reflected as a more accurate aggregated output.

2.3.2 Training

Training the workers on completing the tasks also helps them produce better quality contributions on an individual level. Various techniques have been proposed in the literature to enable this training and most of them occur during task completion. For instance, in Figure Eight's implementation of gold-based quality assurance (test questions) described earlier in Section 2.2.2, when a worker incorrectly answers a test task, a feedback message is shown to explain the error. In addition to controlling the quality, this helps train the workers while they are solving the tasks [9]. Programmatic gold [91] is an extension of the gold-based quality control where test tasks with incorrect answers are also used to train workers against common errors. In this case, test questions - or a subset of them - are injected with known types of errors. By doing this, one gains control over the training experience of the workers and forces all workers to consider most common error cases.

These techniques suffer from the same problem of gold-based quality assurance related to the work load added to the requester side. First, they require manual picking of test tasks which is not scalable. Second, they must provide justified answers for the test question and third one needs to perform a manual audit on a set of completed tasks to be able to detect common errors (this mainly applies for programmatic gold quality assurance). Besides not being scalable, this approach is prone to the lack of data when it comes to new tasks for which the common errors are not known yet.

2.3.3 Task interface design and Priming

Designing the User Interface (UI) of the task is an important step of the task design process. The quality of this interface can have an influence on the quality of the contributions and on the overall crowdsourcing experience. Finnerty et al. [33] studied this influence through a series of experiments where they measured the effects of the complexity of the interface on attention. The results of the experiments showed evidence that a clearer and simpler design (e.g., colors, contrasts, number of tasks in the same interface, ...) and less demand on workers' attention provide more accurate results.

A less common technique that aims to enhance the reliability of a workers is *priming*. Priming as defined by the psychological literature is a temporary, implicit activation of behavioral tendencies as a result of exposure to environmental stimuli [9]. In [84], Morris et

[8] The rational is that when workers get trained on solving the task they will have tendency to stay longer.

[9] Another purpose of this measure is to allow the workers to contest an unfair test question and to rate the quality of the tasks and of the requester in the optional task evaluation step that follows the task completion.

al. showed that it is possible to boost the quality of worker quality by visual priming. That is accompanying the tasks with a visually and emotionally appealing images. Experiments showed that positive priming, i.e., priming with positive content such as a picture of a laughing baby outperformed the negative priming, i.e., priming with negative content such as the picture of a dead body. Worker contributions in both, positive and negative priming scenarios, had a better quality than the neutral scenario where no priming was used.

In [69], Le et al. studied another factor that implicitly influences the contributions of the workers and which we classify under the priming concept. The authors showed that, when quiz tests are used for relevance categorization task, the distribution of the correct answers over the test tasks induces a bias on the worker contributions in the real tasks. For example, if a worker perceives 80% of the answers are of correct answer A, then worker will answer A every time. They concluded that distributing the answers uniformly over the test tasks produces optimal peaks for both the individual worker accuracies and the (MV) aggregated results.

Priming techniques and task design optimization do not give any guarantees on the produced quality. Moreover, priming techniques are very complicated to implement as it necessitates a solid knowledge in human psychology to be able to propose adequate priming to the proposed task type. Optimizing the task design requires the identification of specific design pattern for each task type which might not scale easily due to the growing number of tasks and task types in crowdsourcing systems.

2.4 Multiple stage crowdsourcing

In [28], Dow et al. compared three different workflows occurring between the moment where a worker starts working on a task and the submission of the contribution. First, a classical workflow consisting in working on the task and then submitting it. Second, a workflow where the workers are asked to revise and correct their work after *hav*ing finished working on the task before submitting their contribution. And third, a workflow where the contributions of the workers are reviewed by an external reviewer such as an expert, and returned to the workers for correction before being submitted. This work showed that workers who self-evaluated their work, or have received and external feedback, yielded higher quality contributions when compared to those whose contributions have not been reviewed. However, this work showed also that this quality boost came at the cost of a higher work load for the requester who must determine comprehensive and task specific evaluation criteria that can be used by the workers to self-evaluate their work. Moreover, the reviewing process - especially the expert based one - lengthened the time of task completion as it adds an intermediate step between the work completion and the submission of the contribution. Finally, this adds an additional cost as, naturally, experts must be rewarded for their feedback and workers must be rewarded an additional amount for spending more time on the same task when compared to a more classical workflow.

Zaidan et al. applied in [142] a two stage workflow in the domain of crowdsourced text translation. However, instead of leveraging expert and self-evaluation reviews, they proposed to use peer reviews. That is, the contributions of a worker is reviewed by the other workers. For a given text that needs to be translated, each of the workers to whom the task was

assigned produced a translation of the task. Then, the produced translation was submitted to another set of workers to be edited and rated. Finally, a supervised ranking algorithm ranked the translation based on the grammatical quality of the produced translations to choose the best one. The algorithm uses multiple features to compute the score of a given translation such as the country of residence of the workers, the number of years speaking the needed languages, the language model perplexity of the translation [5], the edit rate of the other translations, and the calibration against professional translators. This approach is specific to translation tasks and its extension to other task types requires a new set of features to compute the scores of the contributions.

To tackle this limitation, Baba et al. [6] propose a similar, yet more generic multi-stage method. Here, reviewers are only allowed to rate the contribution quality (no editing), and the result aggregation method is unsupervised in that no predefined features are needed to assess the quality and compute the scores of the contributions. This is done through an iterative aggregation algorithm similar to the ones described in Section 2.1.2, applied on the reviewer ratings. Here, both the worker ability and the reviewer bias are modeled as latent variable of the generative model. This approach improved the output quality of the crowdsourcing process as compared to aggregating results from a one stage workflow i.e., without reviewing. The authors also showed that for a limited number of workers, the two stages workflow yields better quality results. Indeed, this is an improvement of the quality to budget ratio. Yet, this is limited to unstructured tasks where a two stages process is beneficial such as translation tasks. It does not apply on traditional labeling tasks as one can directly aggregate the contributions instead of aggregating their ratings. Finally, the time overhead related to the sequential stages of the two-stage approach remains unsolved.

2.5 Thesis positioning

After having listed and described the state of the art of quality control in crowdsourcing, in this section we discuss the limitations of methods from a more overhead-related perspective and argue on the positioning of this work.

2.5.1 On the temporal aspect of quality control

Figure 2.2 depicts the projection of major quality control approaches (outlined with boxes) on the timeline of the crowdsourcing process explained earlier in Section 1.1.3 of Chapter 1. One can split this timeline into three zones with respect to the strategic goals of the quality control methods. Those zones are delimited by the red dotted lines. The first zone ends after the task assignment. Methods in this zone try to optimize the access to the crowd wither by preparing the worker before they access the task or by selecting reliable workers. The second zone, which succeeds the task assignment, groups methods that aim at improving the data collection process. That is, the quality of the collected contributions is verified during the collection and bad contributions are dropped or improved on the go. This second zone ends by the end of the contribution collection process after which the third zone starts. This third zone groups the aggregation optimization techniques and aims at maximizing the information gain from the already collected contributions.

This temporal projection of the quality control methods allows us to assess their potential

of coping with the time and cost overhead reduction challenges. In fact, this potential decreases with time. The further a method occurs on the time axis the less potential it gets. Let us consider first the aggregation techniques (Zone 3). Those aim, as mentioned before, to extract the maximal amount of information from the collected contributions. Consequently, they require those contributions to exist in the first place which means that the requester has paid and waited for them to be completed. One can optimize the aggregation methods to require less contributions while performing high accuracy inference of the correct answers. Yet, this is usually limited by the quality of the contributions and the agreement of workers over a given answer which might be harder to achieve for a lower number of workers.

Second, we consider approaches whose the objective is to improve the data collection process (Zone 2). Worker tracking methods i.e., gold-based assurance and fingerprinting, as well as two stage workflows constitute the major part of these approaches. Indeed, intuitively, the potential of these methods in reducing the time and cost overheads is greater than the potential of aggregation methods. That is because the former deals with, yet, uncollected contributions. However, their limitations raise from two different aspects. On the one hand, worker tracking methods track the worker until his accuracy score drops and eliminate him from the campaign. This necessitates the eliminated worker to be replaced which increases the task completion time [19]. Moreover, these workers are not paid neither for test questions nor for the actual task they participated in, which can be discouraging for them[10]. On the other hand, multi stage crowdsourcing lengthens the task completion time as it consists of two phases which occur sequentially[11]. Consequently, methods belonging to this zone tends to have a higher time overhead.

Finally, we consider the approaches that control the input of the task solving process i.e., both the tasks to be submitted and the participating workers. These approaches consist mainly of individual contribution enhancement approaches (Section 2.3) and worker selection approaches [12] (Section 2.2). The former approaches suffer from their dependency on the type of the task. Optimizing the task design is directly correlated to the type of input data, output type, the task solving process and so on. Thus the genericity of this approach and its scalability are not proven. The same applies for priming and worker training since specific per-task priming and training strategies need to be implemented by the requester, which, besides not being generic or scalable, requires additional work and intervention on the requester side. In contrast to this, existing worker selection approaches are more generic and scalable as they require less intervention from the requester side. Some worker selection approaches such as worker filtering, skill matching methods and recommender systems are optimal from a time and a budget overhead point of view since the requester only pays and waits for the needed number of contributions to be completed while having a quality guarantee. However, these methods heavily depend on the individual history of each worker which is not optimal in a system similar to crowdsourcing where tasks are very diversified and worker arrivals and departure are frequent (which leads to a cold start problem for new

[10]Especially for honest workers. In fact, gold-based assurance eliminates malicious workers in a fast manner since their accuracy drops quickly. However, honest workers, who are not skilled enough to complete the tasks might be eliminated after a long journey in the task completion.

[11]A contribution cannot be reviewed before it is submitted.

[12]With the exception of worker tracking approaches.

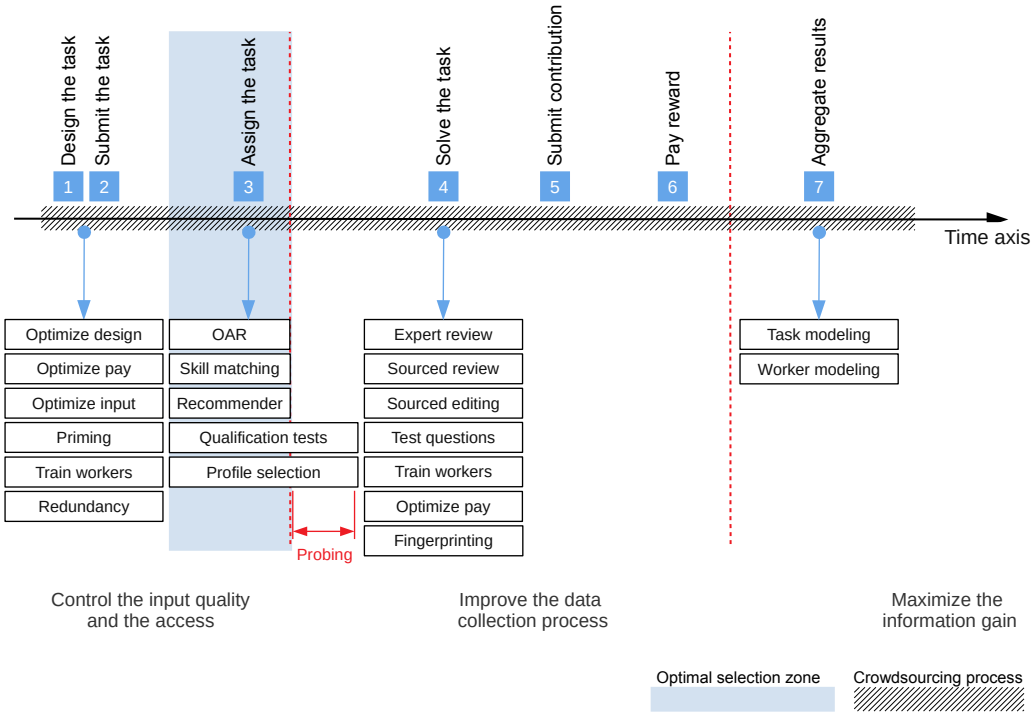


Figure 2.2: A projection of different quality control approaches on the crowdsourcing time line.

workers). Probing based methods such as profile based selection and qualification tests tackle this problem by measuring, in a first round, the reliability of the available workers. During this round, a part of the tasks is distributed to the whole crowd and workers with higher reliability in solving those tasks are selected to continue the work. This allows to estimate the reliability of the worker, per task set, and independently from his historical contributions. Nevertheless, this probing stage consists of paid and time consuming, yet unused, contributions. That is because they are not used, or partially used in the best case, in the actual crowdsourcing but rather for quality control only.

2.5.2 The impact of worker selection on the aggregation process

Selecting workers decreases the time and the cost of completing a high quality crowdsourcing campaign because it limits the task completion to reliable workers without the need for iterative processes such as editing or reviewing. In this section we study the impact of worker selection on the quality of a set of aggregation methods. The objective of this study is to show that by selecting suitable workers, one improves the quality of the final aggregated results. To this end, we created a synthetic dataset detailed in next subsection

2.5.2.1 Dataset

The data generation process is detailed hereafter. First we generated 200 multiple choice question MCQ tasks, each having 4 options. For each of the tasks, we randomly (uniformly) picked one option r among those 4 available ones to be the correct answer of the task. We then generated 500 workers. Each of those workers is characterized by an accuracy $\alpha \in [0,1]$. In our model, α follows a Beta distribution of real parameters $a_1 > 0$ and $a_2 > 0$. We then

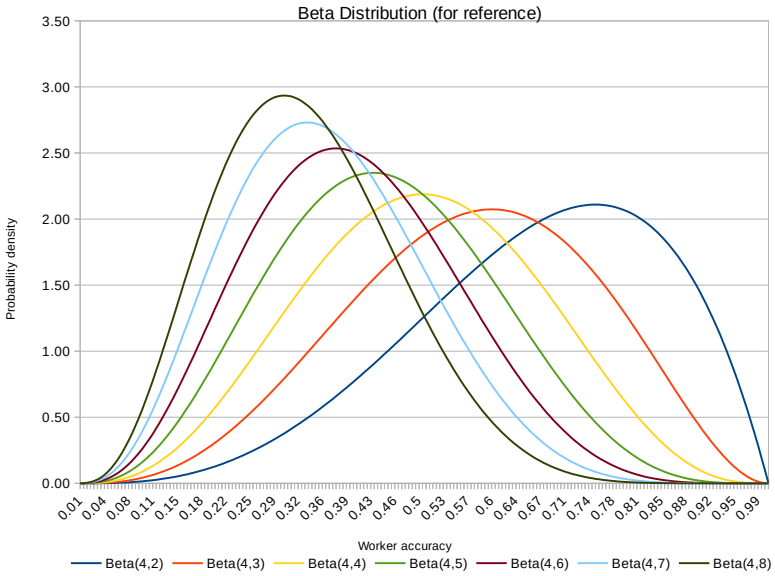


Figure 2.3: The beta distributions used to simulate the worker accuracy distributions with various configurations.

followed the process described in [48] to generate the contributions. That is, the probability of the worker answer being equal to the correct answer r is equal to his accuracy α while the probability of choosing a different answer is equal to his inaccuracy divided by the number of possible wrong options i.e., equal to $\frac{(1-\alpha)}{3}$. This process was repeated 7 times with different configurations for the beta distributions. That is for a fixed value of $a_1 = 4$ and $a_2 \in \{2,3,4,5,6,7,8\}$. This yielded 7 datasets which we denote and index by the value of the corresponding a_2 i.e., D_2, D_3, D_4, D_5, D_6, D_7, D_8. These datasets constituted the input of our evaluation algorithm. Note that the results reported below are averaged results of multiple data generation runs. For each configuration, the datasets were generated 20 times.

The Beta distribution: We chose the beta distribution for the following reasons[13]: first, it is defined in the interval $[0,1]$ which makes it suitable for the accuracy measure. Second, similarly to a normal distribution, it is bell-shaped (for certain range of a_1 and a_2), yet, it is asymmetrical. This means that we can better represent the difficulty of a task from a worker accuracy perspective. Figure 2.3 reports the shape the beta distribution takes with respect to its parameters. a_1 is fixed to 4 and a_2 travels from 2 to 8 with steps of 1. If a task of four options, is extremely difficult, the accuracy of the workers in completing the task is most likely to be close to the accuracy of worker randomly answering the task. That is one divided by the number of options i.e., 0.25 which approximately refers to the top of the bell shape. For this task, workers with very high accuracy (e.g., higher than 70%) and very low accuracy (i.e., close to zero) are very rare. This situation is well modeled with a beta distribution of parameters $a_1 = 4$ and $a_2 = 8$ (dark green line). On the opposite side, the accuracy of the workers for an easy task are most likely to be high. Moreover, very reliable workers are most likely to have an accuracy closer to one. This case is represented by a beta distribution of

[13] A more detailed description of the beta distribution and the intuition behind it have been established by Johnson et al. in [54].

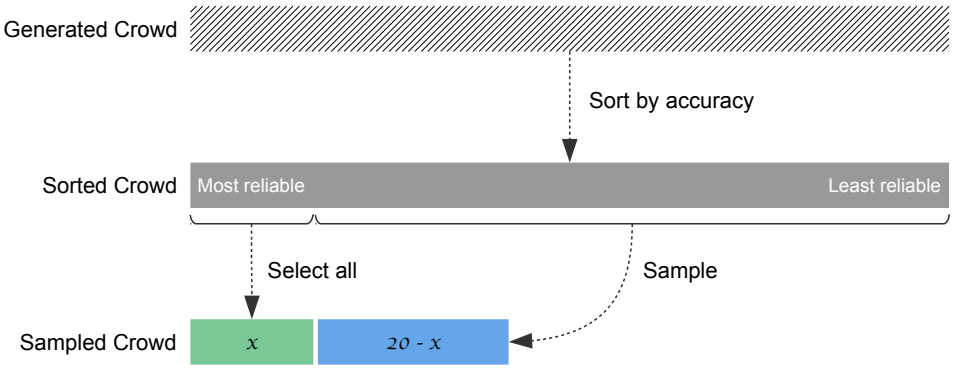


Figure 2.4: The worker sampling process for *Experiment I*.

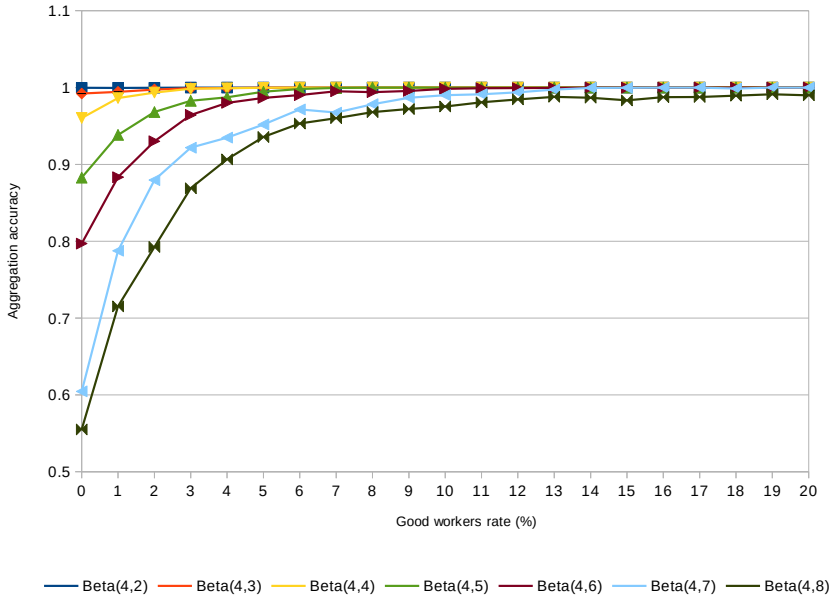


Figure 2.5: The average accuracy of EM in terms of the ratio of reliable workers in the crowd for various worker accuracy distributions.

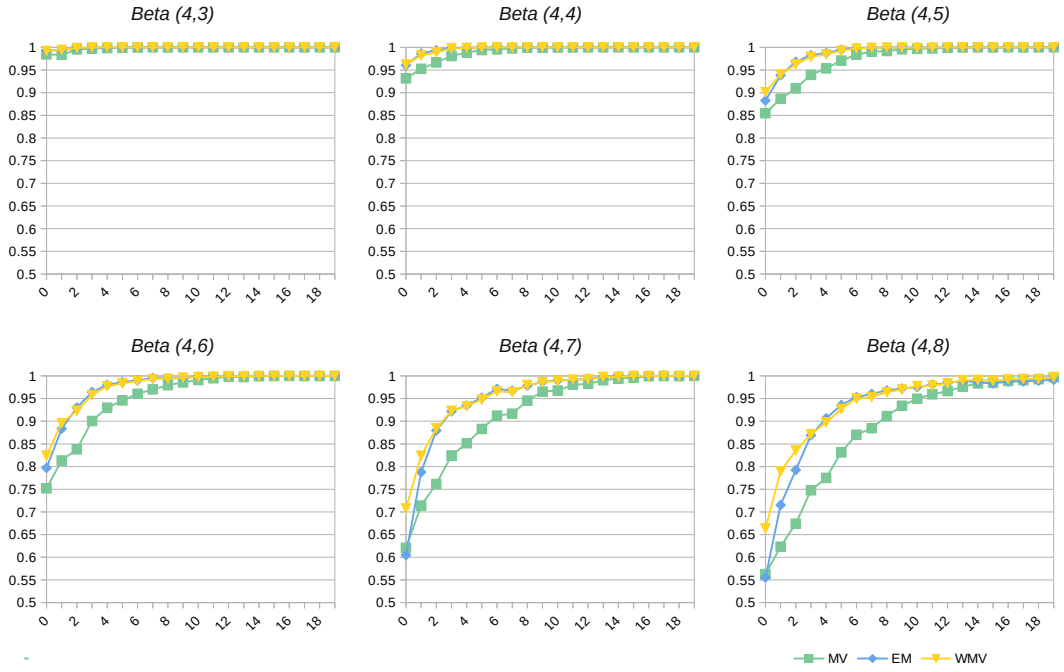


Figure 2.6: A comparison of the behavior of three different aggregation techniques for various distributions of reliable worker (x-axis) in the targeted group of size 20.

parameters $a_1 = 4$ and $a_2 = 2$ (dark blue line). The cases in between represent more or less difficult tasks (light blue, dark red, green, yellow and red lines). In [74, 75], authors also modeled the worker accuracies as a Beta distribution with $a_2 = 2$ and a_1 were computed in a way to keep the average worker accuracy to a given value.

2.5.2.2 Experiments and results

The goal of this study is to show the impact of task difficulty and of the number of reliable workers in the crowd on the output of the aggregation process. We compared three aggregation algorithms as follows: Majority voting (MV), Expectation Maximization (EM), and Weighted Majority Voting (WMV). The latter is based on test questions; for each run the accuracies of the workers for a set of 10 randomly sampled tasks are computed and used as weights for their contributions during the aggregation.

Experiment I

We considered the scenario where 20 answers[14] are collected per task and studied the impact of the reliable worker distribution in the crowd of 20 workers on the aggregation quality. Given a dataset D_i, we picked 20 workers as follows: first, the Top_x workers ($x \in [0,20]$), ranked by their overall accuracy, are selected. They constitute the list of reliable workers. The remaining workers i.e., $20-x$ workers, are then picked randomly from the remaining crowd of size $500-x$. The contributions of those 20 workers for the set of tasks are then aggregated using the aforementioned algorithms, and the accuracy of the aggregation process is computed using the ground truth; it is the average of the correctly guessed answers over the number of estimated answers. This process, depicted in Figure 2.4, was repeated for all

[14] Aggregating the answers from 10 to 20 workers per task is good enough empirically [76, 117].

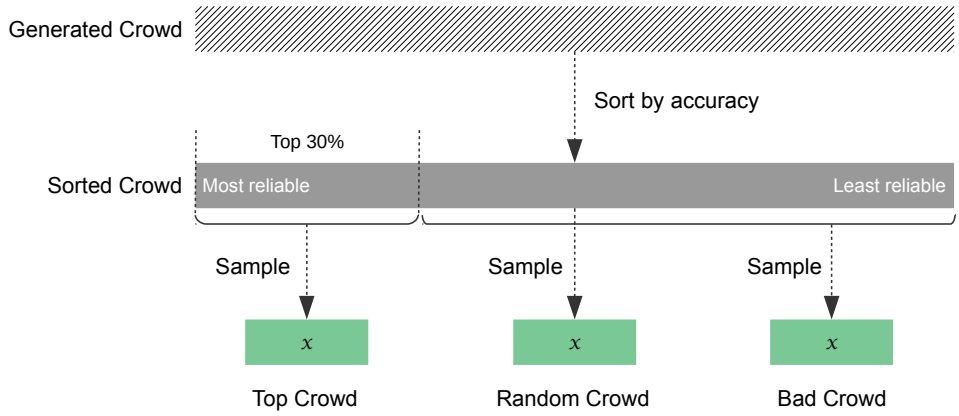


Figure 2.7: The worker sampling process for *Experiment II*.

of the datasets and the results are reported in figures 2.5 and 2.6.

We start by comparing the results of the EM aggregation for the various datasets (Figure 2.5). This reflects the impact of task difficulty on the aggregation process[15]. Clearly the harder the task gets i.e., the higher a_2 is, the lower the performance of EM becomes. That is because for easier tasks, the majority of the workers are reliable, hence, the selection has no real impact on the final quality. Next, we focus on the shape of the curve - which is similar for all of the datasets - and compare the accuracy of EM w.r.t the number of reliable workers x for each dataset individually. It is clear that the accuracy increases by increasing the number x of guaranteed reliable workers in the crowd. With $x = 0$, the 20 workers are randomly picked in the crowd. Hence, the accuracy of the aggregation process is rather low. The more reliable workers we have in the worker set the better is the EM performance. When the 20 workers are reliable, the EM process converges to one regardless task difficulty. These observations and findings apply for the other aggregation techniques as well. This can be seen in the results reported in Figure 2.6. It is worth mentioning that for difficult tasks, both EM and WMV outperformed MV. This held true until the number of reliable workers crossed 12 for Beta(4,7) (i.e., difficult tasks) and 14 for Beta(4,8) (i.e., very difficult tasks). After this threshold all of the aggregation techniques performed similarly. This means that an accurate selection process can eliminate the need for more complicated and time consuming aggregation techniques such as EM. However, achieving a selection process of such quality i.e., nailing the top 14 workers, while possible in a perfect test environment, might not be possible in the real life due to the noise which is not modeled in our synthetic dataset[16]. Yet, results show that the performance improvement is steeper for lower x. That is, a few number of reliable workers is sufficient to consistently improve the quality of the aggregation process. Consequently, a selection process can be advantageous even when it is not perfect.

[15] For the ease of interpretation, the same colors have been used in Figure 2.3 which reflect the difficulty of tasks as discussed earlier and Figure 2.5.

[16] We argue further in this thesis that modeling a realistic noise can be very difficult due to the complexity and uncertainty found in the crowdsourcing environments.

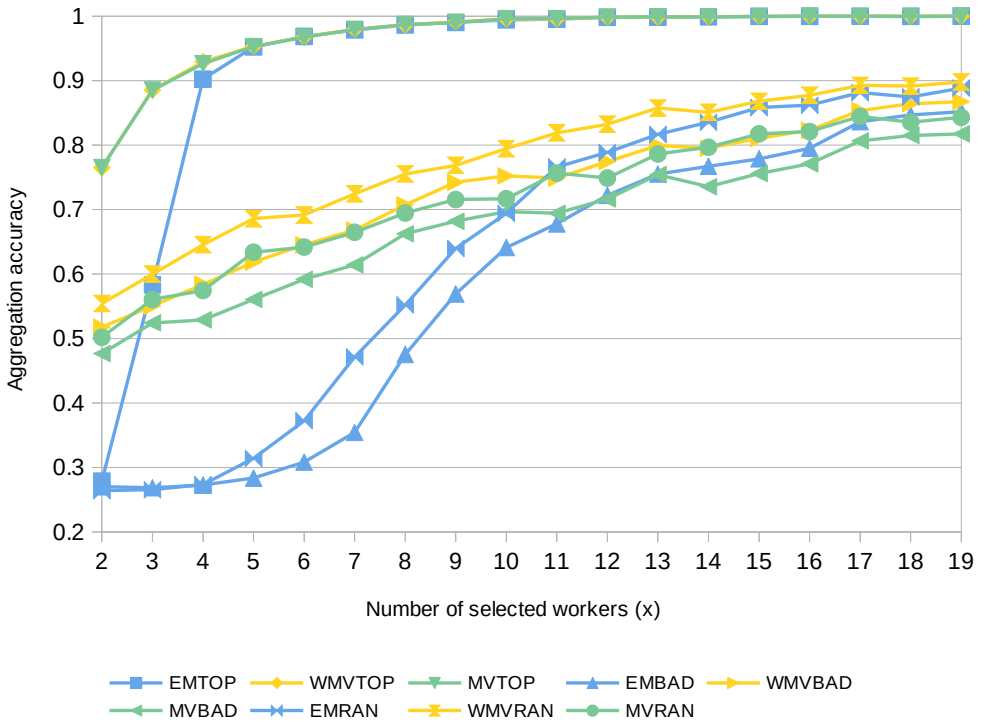


Figure 2.8: A comparison of the behavior of three different aggregation techniques with respect to the number of workers for three different configurations: bad workers, random workers and good workers.

Experiment II

In Experiment I, we studied the quality of the aggregation w.r.t. to the distribution of the reliable crowd in a fixed size crowd, i.e., the number of workers of whom the contributions are aggregated was fixed. Here, we consider the scenario where this number varies and we study the impact of the selection on the aggregation quality. Here we select x workers and aggregate their contributions[17]. In each run, we sampled three sets of x workers from a Beta(4, 4)[18] distribution as follows: (i) Top crowd (TOP) i.e, sampled from the top 30% of the ranked worker list. (ii) Bad crowd (BAD) i.e., sampled from the bottom 70% of the ranked worker list. (iii) Random crowd (RAN) i.e., sampled randomly from the whole population. Figure 2.8 reports the accuracy of MV, EM and WMV for the 3 sampled worker sets. The same findings as Experiment I are observed for this experiments with one exception. In fact, for very low values of x, EM is not able to achieve good accuracy even with reliable workers. This is due to the fact that EM does not converge with very few numbers of contributions per task. Meanwhile, both MV based aggregation techniques were able to achieve better results for the same low numbers of selected workers. Accordingly, one can say that, while costly, redundancy is indeed a key factor of the crowdsourcing process. Hence, restricting the number of selected workers to very low values can be inconvenient.

[17] In contrast with the previous experiment where we aggregated the contributions of 20 workers among which x were reliable.

[18] Task with medium difficulty.

2.5.3 Discussing worker selection strategies

It is clear from the previous discussions that worker selection approaches have the highest potential to deliver a robust quality control process while optimizing its time and cost overheads. However, all selection methods are not equal. Some can be more efficient than others due to three criteria which we enumerate as follows:

A priori This criterion reflects the dependency of a worker selection method on previously collected knowledge about workers. Highly a priori dependent methods, such as recommender based approaches [2, 140] and Overall Approval Rate [32, 35, 123, 138], suffer from cold start and sparse information problems. In contrast to those, methods that are independent from a priori knowledge present either low quality control level such as fingerprinting [108] or the need for time and budget consuming probing stages [76].

Task specificity This criterion reflects how specific the selection process is w.r.t. the tasks. This specificity can be very low, such as selecting workers regardless the type and the content of the task e.g., OAR based selection methods. These methods present no time and cost overheads, yet, their robustness in terms of quality can be questioned since tasks in crowdsourcing are very diverse in terms of required knowledge and skills. On the other side, a selection approach can be very specific such as selecting workers for each submitted task set, e.g., gold-based insurance [68], programmatic gold [91] or other probing techniques [76]. Task-specific approaches guarantee that the selected workers are suitable for the current task which improves the crowdsourcing quality, however, they are time and budget consuming. This is especially true when the same task set is submitted in batches which implies that the same control process will be repeated while the task type and content is unchanged. Methods that rely on describing the tasks as a set of skills [1, 107] are optimal in terms of task specificity. That is because tasks requiring similar skills can be treated similarly even when they are not submitted jointly.

Worker specificity This criterion reflects how specific the selection process is w.r.t. the workers. This specificity can be very low such as generic country based and language based filtering [22]. In this case, there are no guarantees, except for very specific tasks, for the quality of the selected workers since they are judged based on one rough criterion. It can also be very high such as selecting a worker based on his individual history [138] or preferences [3, 18]. In other words, the information used to select one worker cannot be used to select or reject another worker. Methods that rely on workers' declarative profiles [76] are optimal in terms of worker specificity as what is learned from one worker's profile can be applied on other similar profiles.

Table 2.1 summarizes the state-of-the-art of worker selection approaches and allows to compare them based on the aforementioned criteria.

2.5.4 Positioning summary

The temporal projection of quality control methods showed that selection methods offer the highest potential of solving the challenge of time and cost efficient quality control. The

Table 2.1: A comparison of the worker selection approaches in terms of the task specificity, worker specificity and a priori dependency.

Reference	Selection Criteria	Approach	TS	WS	AP
[22, 25, 35, 138]	AOR	Filtering	low	high	high
[26, 69, 91]	Test questions	Tracking	high	high	low
[22, 27]	Qualification tests	Screening	high	high	low
[22, 23]	Country, language	Filtering	low	low	low
[76]	Declarative profile	Probing	high	medium	low
[1, 80, 107]	Skills	Matching	medium	high	high
[96]	Skills and affinity	Group Matching	medium	medium	high
[1, 3]	Preferences	Task composition	medium	high	high
[2, 140]	Preferences	Recommender	low	high	high
[18]	Preferences*	Recommender	medium	high	low
[108]	Behavioral features	Fingerprinting	high	high	low

TS : Task Specificity, **WS** : Worker Specificity, **AP** : A Priori
* From external sources

study of the impact of worker selection on the quality of the aggregation algorithms also proved that this selection is able to improve the output quality of the crowdsourcing process. Yet, studying the literature of selection methods showed that the available selection methods suffered mostly from their dependence on a priori knowledge and from their task specificity. Based on these findings, one can say that an efficient quality control method must adopt a worker selection approach, while eliminating the probing stage and being agnostic to the knowledge of the individual contribution history of the workers. However, this raises the following question : if we eliminate the probing stage and a priori knowledge about individual workers, on which basis should the reliable worker be identified? In this thesis we adopt a worker approach and propose a solution for replacing the probing stage with an a priori agnostic offline learning stage that allows to acquire the needed knowledge about the online crowd to perform high quality worker selection.

2.6 Summary

In this chapter we reviewed the state of the art of quality control methods in crowdsourcing. We highlighted their limitations in terms of quality as well as in terms of cost and time overheads. We argued through a discussion that a worker selection approach that does not require any probing stage might constitute an ideal solution for reducing the time and the cost overhead of the quality control in crowdsourcing. That is because such method would allow to limit the access to the tasks to reliable workers, and thus to collect only trustworthy contributions, while outperforming existing selection methods by eliminating the need for budget and time consuming probing contributions. In the next chapter we present a method called CAWS (Content Aware Worker Selection), which answer this question by leveraging, based on the system history, the relationship between the declarative profiles of workers and their performance in performing tasks of certain types.

3

CAWS : a method for Context Aware Worker Selection

In Chapter 1, we presented the quality problem in crowdsourcing and highlighted the constraints of an efficient quality control approach i.e. the cost, the time and the a priori knowledge. Throughout Chapter 2, we discussed the literature and showed the limitations of existing methods in coping with the quality control challenge under the previously mentioned constraints. The findings of that chapter can be summarized by the following statements: First, despite being reliable in delivering a high quality output of the crowdsourcing process, state of the art methods are unsatisfactory in terms of cost and time overhead. Second, worker selection methods present a higher potential to address this problem as they control the access to the crowd instead of maximizing the throughput of already collected, i.e. paid, contributions. And third, while more efficient, worker selection methods suffer from their dependency on either a priori knowledge about individual workers or budget and time consuming online probing stages.

Based on these findings, we propose in this chapter a worker selection approach called CAWS (Context Aware Worker Selection) to overcome these limitations. CAWS leverages the crowdsourcing system history to build knowledge about the task types and their relationship with the workers profiles. It operates as follows: to begin, in an **offline** phase, completed tasks are clustered into homogeneous groups for each of which the correlation with the workers declarative profile is learned. Then, in the **online** phase (i.e. the actual crowdsourcing), an incoming task is matched to one of the existing clusters and the correspondent profile model is used to select the most reliable workers for the given task.

In summary, this chapter makes the following contributions:

1. We analyze the worker selection problem in crowdsourcing.
2. We describe and formalize a novel worker selection approach which substitutes the online probing stage by an offline learning phase to decrease quality control related cost

and budget overheads and to eliminate the need for prior knowledge about individual workers.

3. We comprehensively discuss the impact of task a grouping technique on the quality of the learning process and prepare the ground for a sound evaluation process to quantitatively assess this impact.

Roadmap. in Section 3.1, we define and formalize the worker selection problem. In Section 3.2, we present CAWS and detail both of its phases and their components and we discuss the impact of the grouping process on the quality of the worker selection and motivate the need for a comparative study. Finally, in Section 3.5 we conclude this chapter and draw the path to the next chapter.

3.1 Setup and problem formalization

Before jumping into the description of CAWS, we start by formally defining the worker selection problem. In this section, the annotations related to tasks and workers are detailed and the worker-selection problem is formalized.

3.1.1 System components

For the sake of clarity we start by defining a crowdsourcing task. In fact, tasks are defined and called differently among the various available crowdsourcing platforms. For instance, while Figure Eight uses *unit* and *job* to denote an individual task and a set of tasks sharing the same description respectively, Amazon Mechanical Turk uses *Human Intelligence Task (HIT)* and *group*. In this work, we define a **task** as follows:

Definition 3.1.1 A **task** is one standalone question or instruction to be answered by a worker. A task consists of (i) an input e.g. an image or a text, (ii) a question or action request e.g. label the image or extract an information, and (iii) an output e.g. free text or an option among proposed answers.

A very common format of task questions is *Multiple Choice Questions (MCQ)* [76]. This is particularly true when considering the dataset labeling scenario stated in Section 1.1.3, as this format is adapted to common dataset labeling tasks such as sentimental labeling of images or text, relevance judgment, data categorization and results validation. Hereafter, we consider only *MCQ* tasks[1] for which we use the following notations :

We denote by t an *MCQ* task in the system.
We denote by $\mathbb{O}_t = \{1, 2, ..., q\}$ the set of q options[a] proposed by the requester for t.
We denote by $r_t \in \mathbb{O}_t$ the solution of t i.e. the only correct option.

[a] Regardless the content of the options, we consider their labels ordered from 1 to q.

Tasks are usually proposed in batches as requesters are rarely interested in answering merely one individual task. Consequently, we define a **job** as follows:

[1] We argue later in this section that this assumption does not impact the generecity of CAWS.

Definition 3.1.2 A **job** is a set of tasks that share the same action request and are proposed by the same requester. A job has a description in which the action to be taken by the worker is described and instantiated.

The set of jobs in the system is denoted $\mathbb{J}$. Figure 3.1 depicts, through an example, a job (a) and an MCQ task (b) as well as an action request (b.1), an input (b.2) and an output (b.3) as defined earlier in Definition3.1.1.

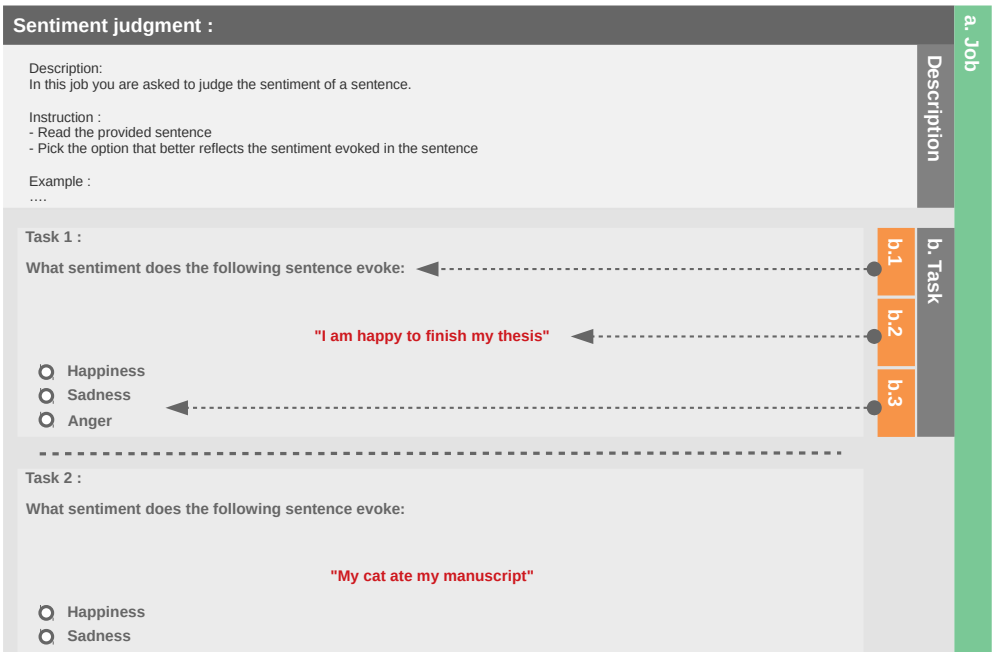


Figure 3.1: The structure of a crowdsourcing job and task.

A crowdsourcing system consists of a set of workers, a set of tasks and a set of contributions. One contribution is the answer given by a worker for a task. That is :

We denote by t a task.
We denote by w a worker.
We denote by $\mathbb{T} = \{1, 2, ..., n\}$ the set of size n of all the tasks in the system.
We denote by $\mathbb{W} = \{1, 2, ..., m\}$ the set of size m of all the workers registered in the system.
The contribution matrix is a $(m \times n)$ matrix C where a coefficient c_{wt} of C denotes the answer given by worker $w \in \mathbb{W}$ for task $t \in \mathbb{T}$

The i^{th} row of C constitutes all the contributions given by the i^{th} workers and the j^{th} column of C constitutes all the contributions given for the j^{th} task. Therefore:

We denote by C_t the column matrix of contributions given for the task t.
We denote by C^w the row matrix of contributions given by worker w.

A worker does not answer all tasks and, consequently, a task is not answered by all of the workers in the system. Hence, C is a sparse matrix where null-value coefficient reflects a non-existing answer. In other words, if a worker w did not answer a task t then, $c_{wt} = null$, otherwise, $c_{wt} \in \mathbb{O}_t$.

3.1.2 TASK AND WORKER CHARACTERIZATION

Task and worker states

In practice, when submitted, a task is not immediately assigned[2] to the workers. We call such a task an *incoming* task.

We denote by $A =< A_1, A_2, ..., A_n >$ the system's assignment vector such that:

$$\text{For each task } t \in \mathbb{T}, \quad A_t = \begin{cases} 1 \quad if \quad \text{t is assigned at least once} \\ 0 \quad if \quad \text{t is not assigned} \end{cases}$$

Consequently:

$$\text{A task } t \in \mathbb{T} \text{ is } \textit{incoming} \iff A_t = 0$$

Naturally, an incoming task does not have any contributions. That is:

$$t \in \mathbb{T} \text{ is } \textit{incoming} \Rightarrow |\{c \in C_t : c \neq null\}| = 0$$

Once assigned, the task steps into the *running* state. A *running* task is an assigned task that still needs all or more contributions until reaching the requested number of contributions set by the requester and which we denote ρ_t.

$$\text{A task } t \in \mathbb{T} \text{ is } \textit{Running} \iff A_t = 1 \wedge |\{c \in C_t : c \neq null\}| < \rho_t$$

When all the requested contributions have been gathered and no more contributions can be submitted, a task promotes to the *completed* state:

$$\text{A task } t \in \mathbb{T} \text{ is } \textit{Completed} \iff A_t = 1 \wedge |\{c \in C_t : c \neq null\}| = \rho_t$$

Depending on his availability on the platform, a worker might have two states. He is *online*, if at the moment of the task submission, he is connected to the platform and ready to solve tasks, otherwise, he is *offline*. Accordingly:

We denote by $\mathbb{W}_o \subseteq \mathbb{W}$ the set of online workers.
We denote by $\overline{\mathbb{W}}_o$ the set of offline workers. That is the complementary set of $\mathbb{W}_o$ in $\mathbb{W}$:

$$\overline{\mathbb{W}}_o = \{w \in \mathbb{W} : w \notin \mathbb{W}_o\}$$

Task and worker features

We consider that a task is characterized by a feature vector of size r. We denote it V_t:

$$\text{For } t \in \mathbb{T}, \quad V_t =< V_t^{(1)}, ..., V_t^{(r)} >$$

The vector V_t can reflect different representations of the task such as a TF-IDF representation, a Doc2Vec embedding, a skill set, a taxonomy instance, etc.

Similarly, we consider that each worker is characterized by a feature profile of size s which we denote P_w:

[2]The assignment might be performed by the platform through selection or by the workers through self assignment i.e. choosing the task from a list of proposed ones.

$$\text{For } w \in \mathbb{W}, \quad P_w = < P_w^{(1)}, ..., P_w^{(s)} >$$

A worker profile has three types: (i) a *declarative* profile built using information provided by the user, (ii) a *derived* profile computed from the user interaction in the system and (iii) a *hybrid* profile which combines both types of data. In our terminology we consider the first type hence P_w denotes the **declarative** profile of w. Finally, we assume that the performance of each worker, in completing a set of tasks, is measured by how often his answer was correct. That is, the proportion of correctly answered tasks out of all the answered ones in the given set. We denote this accuracy α and formally express it as follows:

$$\alpha_w^{\mathbb{G}} = \frac{1}{|\{c_{wt} : c_{wt} \neq null, \forall t \in \mathbb{G}\}|} \sum_{t \in \mathbb{G}} I(c_{wt}, r_t) \tag{3.1}$$

Where w is a worker, $\mathbb{G} \subseteq \mathbb{T}$ is a task set, $\alpha_w^{\mathbb{G}}$ his accuracy for the tasks in $\mathbb{G}$, C^w his contribution vector, and I is an indicator function such that:

$$I(x,y) = \begin{cases} 0 & \text{if} \quad x = y \\ 1 & \text{otherwise} \end{cases} \tag{3.2}$$

Here, we use an indicator function since it is adapted to *MCQ* tasks. The same accuracy measure can be used for different task types where the output space is different e.g., multiple correct answers are possible, ratings, free text, etc. Yet, in these cases the indicator function must be replaced by a more suitable function to assess the correctness of the worker answer. For instance, in the cases where multiple answers are possible for the same question, I can return the precision of the worker in which case the performance of a worker for a set of tasks is his average precision. Accordingly, for a given task and regardless the type of its output, it is possible to compute a similar measure to asses the performance of a given worker in achieving it. Therefore, considering only *MCQ* tasks, which helps simplify the formalization of the problem, does not affect the generecity of our approach. It is worth noting that, aside from the indicator function, the other elements of the formalization are independent from the choice of the task type i.e. *MCQ*.

3.1.3 The worker selection problem

Assume that a function $f_{C,\tau}$ which, knowing the system history C, allows to estimate the accuracy of the workers for a given task τ exists:

$$f_{C,\tau} : \mathbb{W}_o \to [0,1], w \to \hat{\alpha}_w = f_{C,\tau}(w)$$

The worker selection problem consists in selecting, for a given task τ, the λ workers in $\mathbb{W}_o$ with the highest estimated accuracies in completing τ. Here, λ is a configurable parameter indicating the number of workers to select. That is:

Let S be the selecting function:

$$S_\lambda(\tau) = Top_\lambda(\{f_{C,\tau}(w); \quad w \in \mathbb{W}_o\})$$

The challenging part of the worker selection problem is to find the function f that optimally estimates the workers' accuracies which consequently helps infer the correct answer r_t for a given task t with higher confidence for a given aggregation function.

For convenience, a summarized view of the annotations described earlier, alongside with their significations can be found in Table 3.1.

Annotation	Description
$\mathbb{T}$	The set of tasks in the system
$\mathbb{J}$	Set of jobs in the system
$\mathbb{W}$	The set of workers in the system
$\mathbb{W}_o$	The set of online workers
$C_{(m\times n)}$	The contribution matrix
t	A task
w	A worker
j	A job
n	Number of tasks in the system
m	Number of workers in the system
C^w	Contribution vector for worker w (row)
C_t	Contribution vector for tasks t (column)
ρ_t	Requested contributions for task t
r_t	Correct answer of task t
V	Task feature vector
P	Worker profile
α_w	Accuracy of worker w

Table 3.1: A summarized view of the annotations used in this chapter.

3.2 Context Aware Worker Selection

3.2.1 Vision

We discussed in the previous chapter the limitations of existing quality control methods. These limitations mainly orbit around two major issues: the budget and time of performing quality control and the knowledge needed to complete it. Clearly, by collecting a large number of contributions per worker or per task, the aggregation process achieves a more accurate inference of the task answers [48]. Yet, this requires a higher budget and longer time before the task is completed. Another approach consists in selecting fewer, yet better quality, contributions. By only allowing a selected set of workers to answer a given task, the time and budget overheads resulting from the quality control process can be better optimized. That is because only the needed contributions are paid. Nonetheless, this requires the presence of some knowledge allowing the selection of suitable workers. In the literature, this knowledge is inferred in various ways such as using the overall approval rate (OAR) of the workers (Section **??**), using worker screening and qualification tests (Section 2.2.1) or through online paid probing (Section 2.2.4). Using the OAR assumes that all workers have already

enough contributions (otherwise, it cannot be representative of the worker's performance) and ignores the particularity of the incoming task. Worker screening is performed before the actual crowdsourcing, which means that workers are not obliged to contribute correctly once elected upon the screening process. Test-question-based selection requires a set of tasks with justified answers[3], which implies an additional work load on the requester side and makes the selection prone to requester bias in the provided gold answers. Moreover, test questions are distributed over a long period within the job solving process[4], which means that some workers might spend a long time solving the tasks before getting eliminated from the job because of a quality drop. On one hand, this increases the time of job completion as eliminated workers must be replaced and, on the other hand, this is unfair regarding the time the worker spent doing the job without being paid. Finally, probing stages are done online and tasks solved during them are paid which is not optimal from a time and a budget perspective.

In order to cope with the aforementioned knowledge source problem, we propose to provide this knowledge about workers from the contributions of previously completed tasks. That is, by learning the performance of workers from their historical contributions. Yet, in order to avoid falling into a cold start problem i.e. workers with few or no historical contributions, workers are seen by their declarative profile. In other words, the selection is achieved based on a declarative profile instead of a worker himself. Indeed, it has been shown in the literature that a correlation between some profile features and the profile of the workers exists (Section 2.2.4).

Recall : Kazai et al. [61] showed that a correlation between the performance of the workers and some of their demographical and personality-related features exists. Li et al. [76] uses the declarative profiles to improve the selection. Jin et al. [53] optimizes the aggregation using information such as demographics and domain-related self appraisal.

In the remainder of this chapter, we show how we leverage this correlation toward an cost- and time-efficient, a-priori-agnostic selection of workers.

3.2.2 Method overview

The general workflow of the method which consists of two steps - depicted in Figure 3.2 - can be summarized as follows: in the *offline learning phase* (Figure 3.3 - A), a feature vector is extracted for each task in the history (step A.1), then tasks are clustered based on these vectors (step A.2). For each cluster, the vector of worker features that maximizes the contribution quality is determined (step A.3) and stored (step A.4). In the *crowdsourcing phase* (Figure 3.6 - B), the features of the incoming task are extracted (step B.1) and used to match the task to one of the existing group of tasks (step B.2). The feature vector associated to the found type during the learning phase is fetched (step B.3) and used to estimate the workers' accuracies for the current task (step B.4). Workers with higher accuracy are then selected to contribute to the task. In the next two sections, the steps of the offline learning and online targeting phases are described.

[3] On the Figure Eight platform, most jobs have between 50 and 100 test questions [26].

[4] For instance, by default, Figure Eight uses 1 test question per task page i.e. 1 to 19 tasks [26].

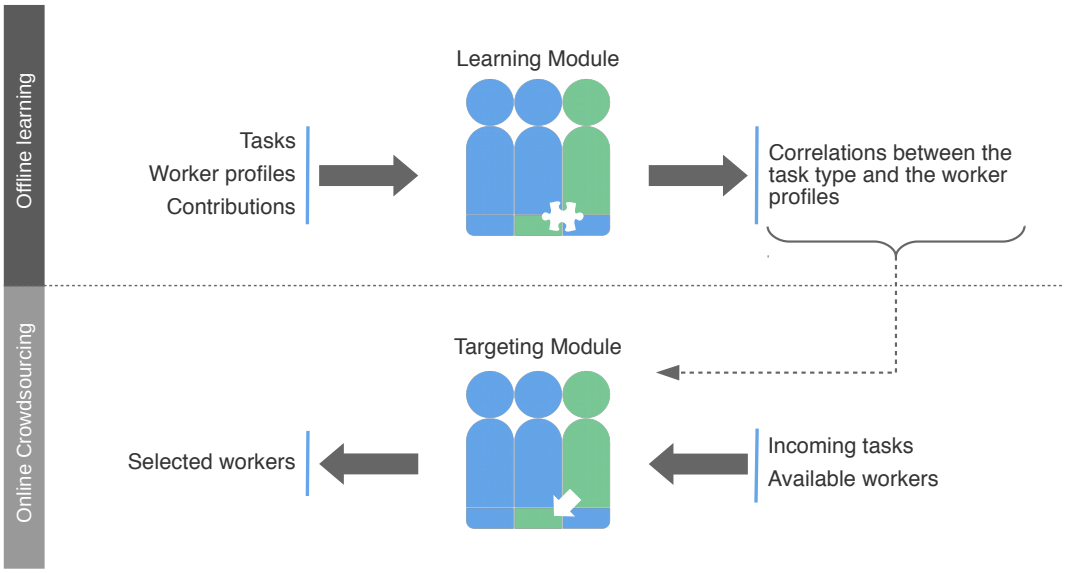


Figure 3.2: An overview on the workflow of CAWS.

Practical requirement

CAWS is meant to require as little human input as possible. Indeed, crowdsourcing systems are highly dynamic in terms of tasks and worker arrivals and departures, which prevent the possibility of manually performing the task characterization or the selection process [37]. As mentioned earlier, to deal with the cold start problem, we only rely on the declarative profile to find and target the reliable workers. Similarly, we only use features extracted from the task documents to characterize the tasks. We do not use any prior characterization made by the requester. This is mainly because such characterization might be imprecise, very narrow, very generic etc. Consequently, worker profile features and task features belong to different spaces [5]. Thus, explicit feature matching cannot be used to select workers.

3.3 Offline Learning

3.3.1 Task Grouping

The first step of CAWS consists in splitting the task set of the crowdsourcing system into homogeneous groups to which we refer as "Contexts". The task grouping consists of two steps: (1) feature extraction and (2) task clustering.

1. Task vectorizing (A.1)

This step consists in extracting the features of the tasks. We consider a task as a text document and produce its vectorial representation. To this end, we propose to use either of two document vectorization techniques: one term frequency based technique called TF-IDF (Term Frequency Inverse Document Frequency) [143] and one word embedding technique called Doc2vec (Document to Vector embedding)[70]. Indeed, other techniques exist to compute a vector representation of a text document. An example of these techniques are

[5] In contrast with task and worker modeling adopted in some skill-based matching techniques [107]. In these case, both the tasks and the workers are characterized by a set of qualifying skills.

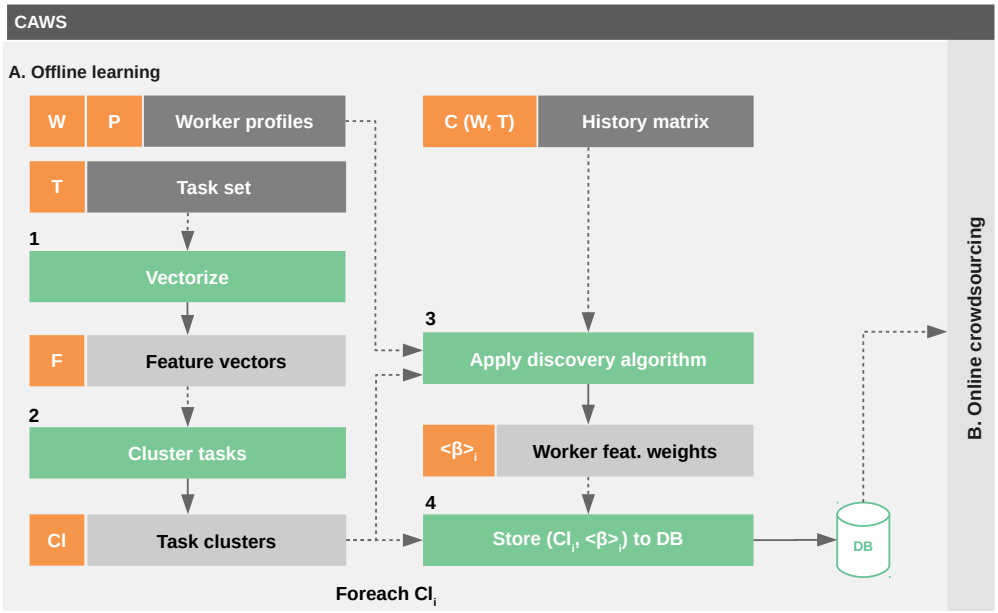


Figure 3.3: An overview of the offline learning phase of CAWS.

topic modeling techniques such as Latent Semantic Analysis (LSA) [29], Probabilistic Latent Semantic Analysis (PLSA) [45], Latent Dirichlet Allocation (LDA) [11] and LDA2vec [6]. These techniques are suitable for fairly long text documents or for short documents with few dense topics such as tweets [87, 130]. Yet, none of these is the case for crowdsourcing tasks. Note that TF-IDF and Doc2vec have been also used in the literature to characterize crowdsourcing tasks such as in [8].

2. Task clustering (A.2)

Once vectorized, the tasks are clustered using one of two clustering technique: K-means and agglomerative clustering. This step yields a set of task clusters, each of them defining a task context. The set of all clusters is denoted *Cl* and it constitutes, along with the system contribution matrix and the worker set, the input of the next step i.e., the performance inference.

The impact of the grouping process : A discussion

Depending on the grouping technique used to split the task space, the contexts would emphasize certain kind of similarities between the tasks. To clarify this idea, we illustrate it by the simplified example depicted in Figure 3.4. For the same task set $\{t1,t2,t3,t4\}$, different grouping methods can yield different distributions of the tasks over the contexts. For instance, if the used grouping emphasizes the action to be performed during the task solving (i.e., the feature "Recognize" presented in green), all the tasks would be part of the same context (*Grouping 1*). The same reasoning can be maintained with the second feature presented in blue, which in practice, reflects the actual knowledge stimulated by the task. Considering this feature would burst split the task set into four disjoint contexts (*Grouping 2*). Finally, if the type of the input media is considered, then, the grouping would produce

[6] An extension of word2vec and LDA that jointly learns word, document, and topic vectors.

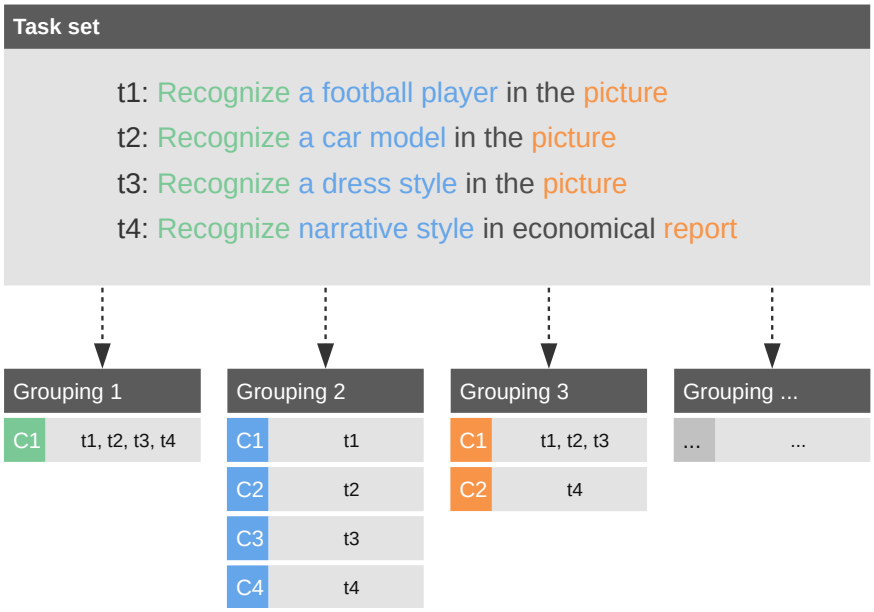


Figure 3.4: An illustrative example of the grouping importance.

two contexts relative to the features presented in orange i.e., one context for picture and one context for report (*Grouping 3*). In the discovery step - explained in detail later in this chapter - CAWS infers the correlation between the worker profile and his contextual performance. That is, it determines which profile features are most likely to impact the worker performance for the considered context. Consequently, the more the contexts are significant from a worker feature perspective, the more reliable is the computed correlation. For example, consider the tasks of Figure 3.4. Let us suppose that each worker w is characterized by a three-feature-profile $P_w =< Gender, Education, Interest >$. Clearly none of these three features can be significant to identify more reliable workers for completing all the tasks in context *C1* resulting from *Grouping 1*. That is because none of them can explain why one worker could be better - on average - than another worker in completing all of these tasks. Meanwhile, the feature "interest" and/or "gender" can be very significant to identify workers who are reliable in completing tasks for each of the contexts *C1*, *C2*, *C3* and *C4* resulting from *Grouping 2*. That is, on average, a worker with *interest* = *"sport"* and *gender*= *"male"* for instance would be more reliable in contexts *C1* and *C2* while workers with *interest* = *"Reading"* would be more reliable for context *C4* and so on.

Furthermore, the produced contexts depends on the combination of the vectorizing and clustering algorithm. The study[7] depicted in Figure 3.5 compares the output of a simple clustering task[8] with respect to the distribution of the data points in the feature space. It is clear that different clustering algorithms can yield different grouping for the same data point distribution (column-wise comparison). Moreover, the output quality of a given clustering algorithm can vary depending on the data the algorithm is dealing with (row-wise

[7]Modified from the original study found on : `http://scikit-learn.org/stable/modules/clustering.html`.

[8]This study is merely an illustrative example of the dependence between the clustering and vectorizing algorithms. Data used in this study are 2-dimensional which is not necessarily true in the case of tasks.

comparison).

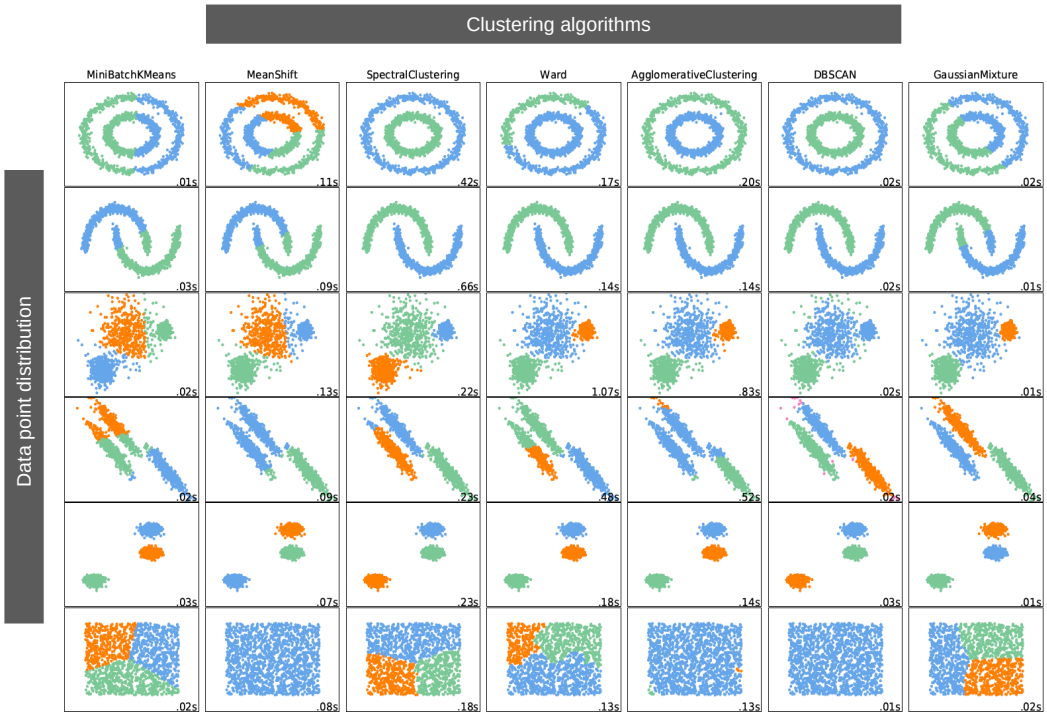


Figure 3.5: An illustrative example of the behavior of different clustering algorithms (columns) w.r.t various data point distribution (rows).

In summary, grouping the task is an important, yet tricky, step. We show further in this thesis how the choices of the vectorizing and clustering algorithms are made so that the learning phase yields exploitable results.

3.3.2 The Discovery Algorithm

The step that follows the task grouping consists in inferring the worker features that maximizes the workers' performance for each type. For this purpose, an algorithm similar to the discovery algorithm described in [76] is used. It consists of two steps. First, a *performance inference* step in which the workers' individual accuracies are computed. Second, the *beta vector inference* step, in which these accuracies are used as targets for a linear regression model, of which the observations are the workers' profiles, to find the most significant worker features. In contrast with the work proposed by Li et al. in [76] where the discovery algorithm is performed on every incoming job online, CAWS applies it offline and by cluster i.e., by task type. When applied on all the clusters in the system, the discovery algorithm yields a set of profile models, each corresponding to one task type. Hereafter, we call the profile model computed for a given cluster the cluster's *perfect profile*. Following are the details of the two steps of the discovery algorithm.

Performance inference

Offline learning deals with tasks that have been already completed, hence, we assume that the correct answers have been estimated during the aggregation process. In other words,

for each *completed* task t, we assume that the correct answer r_t is known. Consequently, computing the accuracy of a given worker for the completed tasks is possible using Equation 3.1. However, here, we are interested in the contextual accuracy of a worker. That is, his accuracy for all the tasks he completed in a given task cluster.

Let $cl \in CL$ be a task cluster and $w \in \mathbb{W}$ be a worker, the accuracy of w in cl, denoted α_w^{cl}, is expressed using Equation 3.1 as follows:

$$\alpha_w^{cl} = \frac{1}{|\{c_{wt} : c_{wt} \neq 0,\ t \in cl\}|} \sum_{t \in cl} I(c_{wt}, r_t) \tag{3.3}$$

Equation 3.3 supposes that all the workers have contributed to, at least, one task belonging to cl, which is not realistic. Therefore, we define the cluster crowd which is the set of workers that contributed to at least one task in cl. We denote it $\mathbb{W}_{cl}$ ($\mathbb{W}_{cl} \subseteq \mathbb{W}$) and we express it as follows:

$$\mathbb{W}_{cl} = \{w \in \mathbb{W} : \exists t \in cl,\ c_{wt} \neq 0\} \tag{3.4}$$

Output of the step : Accordingly, for the considered cluster cl, the output of this step is a vector of size ($|\mathbb{W}_{cl}| \times 1$). We denote this matrix α^{cl}, and we express it as follows:

$$\alpha^{cl} = (\alpha_1^{cl} \quad \alpha_2^{cl} \quad \dots \quad \alpha_{|\mathbb{W}_{cl}|}^{cl}) \tag{3.5}$$

Beta vector inference

The second step of the discovery algorithm consists in determining the worker features which are correlated with a good performance in completing the tasks of a given cluster. We use the linear regression model as shown in Equation 3.6. Let w be a worker, P_w his profile, α_w^{cl} his accuracy in a cluster cl and ε a Gaussian noise with mean 0:

$$\alpha_w^{cl} \sim \beta_0 + \beta_1 P_w^{(1)} + ... + \beta_p P_w^{(p)} + \varepsilon \tag{3.6}$$

Hence, for a complete cluster cl:

$$\begin{pmatrix} \alpha_1^{cl} \\ \alpha_2^{cl} \\ \vdots \\ \alpha_{|\mathbb{W}_{cl}|}^{cl} \end{pmatrix} = \begin{pmatrix} 1 & P_1^{(1)} & \dots & P_1^{(p)} \\ 1 & P_2^{(1)} & \dots & P_2^{(p)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & P_{|\mathbb{W}_{cl}|}^{(1)} & \dots & P_{|\mathbb{W}_{cl}|}^{(p)} \end{pmatrix} \times \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{pmatrix} + \begin{pmatrix} \varepsilon_0 \\ \varepsilon_1 \\ \vdots \\ \varepsilon_{|\mathbb{W}_{cl}|} \end{pmatrix} \tag{3.7}$$

Fitting the model of Equation 3.6 yields the estimated values $\hat{\beta}_i$ of the feature weights β_i. These weights reflect the relative importance of each profile feature in influencing the worker performance. The set of weights computed for a given cluster cl form the *Beta-vector* of cl and is denoted $\hat{\beta}_{cl}$.

Algorithm 1: The learning algorithm

Input : System history C
worker set W
task cluster $cl \in Cl$

Output : Estimated correlations for the cluster $(cl, <\hat{\beta}_{cl}>)$

1 **Function** *learn(cl)* **:**
2 $W_{cl} \leftarrow \{w \in W \mid \exists t \in cl, C_{wt} \neq \varnothing\}$ // Workers who answered at least one task in cl
3 **foreach** *w in* W_{cl} **do**
4 $\alpha_w^{cl} \leftarrow computeAccuracy(C, w, cl)$ // Equation 3.3
5 $\alpha_{cl}.add(\alpha_w)$
6 $<\hat{\beta}_{cl}> \leftarrow fit(\alpha_{cl}, W_{cl})$ // Equation 3.6
7 **return** $(cl, <\hat{\beta}_{cl}>)$

Output of the step : The discovery algorithm is applied on every task cluster apart (See function *learn()* in Algorithm 1). Hence, the overall output of the offline learning phase is a set A of *(cluster, Beta vector)* couples. A is expressed as shown in Equation 3.8.

$$A = \{(cl, <\hat{\beta}_{cl}>), cl \in Cl\} \text{ where } <\hat{\beta}_{cl}> = <\hat{\beta}_0, \hat{\beta}_1, ..., \hat{\beta}_p>_{cl} \tag{3.8}$$

3.4 Online crowdsourcing

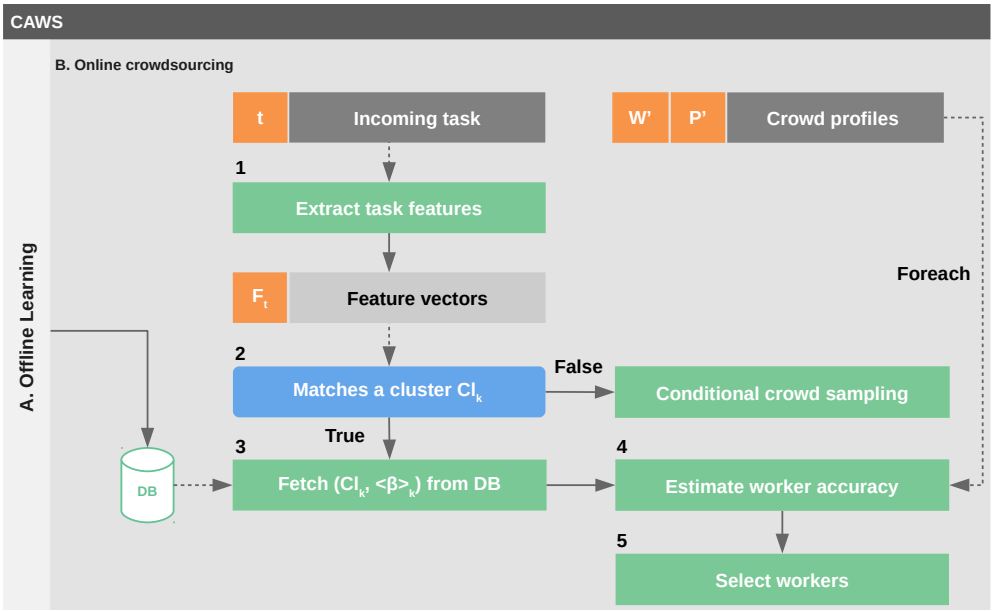


Figure 3.6: An overview of the online crowdsourcing phase of CAWS.

The online crowdsourcing phase is depicted in figure 3.6 - B and detailed in Algorithm 2. For a given incoming task, selecting the reliable workers is done as follows: first, the features of the task are extracted (*line 2*), then the feature vector is matched to an existing context.

Algorithm 2: Targeting Algorithm

data : Incoming task τ
online workers W_o
learned associations A
selection parameter λ
cluster list Cl

Output : selected worker set W_s

1 **Function** *target*(τ, W_o, A, λ) :
2 $F_\tau \leftarrow extractFeatures(\tau)$
3 $cl_\tau \leftarrow argmin(distance(Center_{Cl}, F_\tau))$ // find the cluster to which τ belongs
4 $< \hat{\beta}_{cl_\tau} > \leftarrow \{< \hat{\beta} > | (< \hat{\beta} >, cl) \in A \text{ and } cl = cl_\tau\}$ // fetch the beta-vector associated with the current cluster
5 **foreach** w *in* W_o **do**
6 $\hat{\alpha}_w \leftarrow dotProduct(w.P, < \hat{\beta}_{cl_\tau} >)$ // estimate the accuracy of w using his profile w.P
7 $W_o^{Sorted} \leftarrow W_o.sortBy(\hat{\alpha})$ // rank wortkers by accuracy
8 **for** i *in* $0 \rightarrow \lambda$ **do**
9 $W_s[i] \leftarrow W_o^{Sorted}[i]$ // select top wortkers w.r.t. λ
10 **return** W_s

The matching consists in finding the cluster with the nearest centroid to the task feature vector (*line 3*). Afterwards, the online workers are ranked with respect to their estimated accuracy in completing the task computed using the model of Equation 3.6, their profiles and the learned associations A for the matched type (*lines 5-7*). Finally the most reliable workers are selected to complete the task (*line 8-10*). In this algorithm, k is a *selection parameter* that determines the number of workers that should be selected. This parameter reflects the requester's needs in terms of budget and quality.

3.5 Summary

In this chapter we proposed and formalized CAWS, a Context Aware Worker Selection method for crowdsourcing that aims at reducing the time and the budget overheads. To this end, CAWS substitutes the online probing stage found in state of the art approaches by an offline learning process which is transparent to the requester from both a budget and a time perspectives. In summary CAWS operates as follows: offline, tasks in the system history are clustered into homogeneous groups, for each of which a correlation model with the worker profile features is learned. Then, online, the learned model are used to select reliable workers in the crowd in a context aware manner. That is, for a given incoming task, the model learned for its type will be used.

We argued in this chapter, that relying on the declarative profiles of the workers, is beneficial to address the cold start problem which is inherent to methods that are based on individual overall-approval-rate. These are indeed widely adopted by commercial platforms.

Then, we demonstrated through a simulation study that the cost and time overhead can be significantly reduced by moving crowd probing into an offlline phase. Yet, while reducing the overheads and being agnostic to a priori knowledge about individual workers constitutes the constraining criteria of this work, improving, or at least maintaining, the throughput of the quality control process is crucial. Thus, evaluating the performance of CAWS in terms of selection accuracy is mandatory.

While evaluating aggregation-based, skill-matching-based and qualification-based quality control approaches is feasible using synthetic datasets and datasets lacking for worker and task meta-data as well, evaluating CAWS is more demanding in terms of data. On one hand, CAWS leverages a correlation[9] that is not realistically reproducible in synthetic datasets. On the other hand, it exploits workers and task informations usually absent from the existing ones. Hence, the challenge of designing and building a suitable evaluation dataset should be dealt with. In the next chapter, we tackle this challenge by presenting a new crowdsourcing evaluation dataset to prepare a sound evaluation process.

[9]The correlation between worker profile and task type.

4

CrowdED and CREX : Enabling Quality Control Evaluation

In Chapter 3, we described a selection method called CAWS that aims at reducing the budget and the time overheads resulting from the quality control process while improving its yield. In order to measure this quality and overhead gain, a two-fold evaluation process is needed. The first fold consists in measuring the budget and time gained by shifting the learning process to an offline phase, and the second involves measuring the impact of the task clustering on the quality of the learning and - consequently - of the targeting (Sections 3.3 and 3.4). Performing this evaluation raises a dataset-related challenge unforeseen in the literature. In fact, none of the existing worker selection methods leverages, at the same time, the declarative information of the workers and the content of the tasks to optimize the selection step. Consequently, existing datasets - tailored to evaluate these methods - do not contain these data simultaneously. Since CAWS exploits both of the aforementioned data, it is clear that existing datasets are not suitable to perform the required evaluation. To address this challenge we designed and collected CrowdED (Crowdsourcing Evaluation Dataset) an information-rich evaluation dataset. In this chapter, we detail and motivate the creation of CrowdED and we describe CREX (CReate Enrich eXtend), a platform that allows the collaborative extension and enrichment of CrowdED.

The contributions of this chapter are:

- We provide a comparative review of the existing datasets and discuss their usability in our evaluation process based on a comprehensive set of requirements.
- We propose CrowdED, a rich evaluation dataset of which we present the design and the contribution collection steps as well as the statistical and structural properties.
- We propose a sampling algorithm to create budget constrained task corpora.

- We assess the contribution of CrowdED in closing an important dataset gap in the crowdsourcing community through a qualitative study.

- We present the design of CREX and show how it facilitates the creation of crowdsourcing campaign to extend and enrich CrowdED.

Roadmap. In Section 4.1, the requirements of the evaluation process are analyzed and the specifications of a suitable dataset are set. In Section 4.2, state of the art crowdsourcing evaluation datasets are discussed w.r.t the requirements stated earlier. Then, in Section 4.3, we describe the creation process of CrowdED as well as its structural and statistical characteristics. Finally, we present CREX in Section 4.4, before summarizing this chapter in Section 4.6.

Acknowledgment. In this work, the paid crowdsourced data collection step has been made possible thanks to the financial support of the Franco-German University DFH-UFA - `https://www.dfh-ufa.org/`

4.1 Objectives and requirements

The first step towards perform the two-fold evaluation stated earlier is to select a suitable evaluation dataset which in turn starts by setting its qualitative specifications. To this end, we analyze the requirements of our method and deduce the specifications of a suitable dataset.

S1 : Data richness

As discussed in the previous chapter (Section 3.2.2), CAWS tackles the time and budget reduction challenges by replacing the online probing and worker screening phases by an offline learning step. CAWS leverages, on one hand, the similarity between the tasks in the history and the incoming tasks and, on the other hand, the workers' declarative profiles. Therefore, a suitable evaluation dataset should contain the information that can allow one to assess the similarity between historical and incoming tasks as well as the worker profiles. Based on this reasoning, we distinguish two specifications related to the richness of the dataset:

S1.1 The dataset must provide information about workers. That is, the workers' declarative profiles.

S1.2 The dataset must provide information about tasks. That is, their full content i.e. description, questions and answer options.

S2 : Data diversity

Crowdsourcing tasks cover a wide range of types [111]. Similarly, workers in a crowdsourcing system fall into multiple profile groups [52]. In order to allow assessing the genericity of our proposition, it is crucial that the evaluation dataset reflects - to a sufficient extent - this type and profile diversity. It is worth mentioning that while qualitative studies on the crowdsourcing tasks and workers exist [52, 61, 105], quantitative studies describing the distribution of these tasks and workers, over their types and features respectively, are lacking in the literature. Hence, it is hard to study whether a dataset is statistically representative of

the crowdsourcing system or not. Accordingly, we set two specifications related to the data diversity :

S2.1 The dataset must reflect the diversity of the profile features characterizing the workers of a real crowdsourcing system.

S2.2 The dataset must reflect the diversity of the task types. This includes the generic asked action e.g. labeling an image, judging relevance, analyzing sentiment in text, etc., and the actual knowledge domain of the task e.g. sport, economy, botany, etc.

It is clear that Specification **S.2** tightens the Specification **S.1**. Indeed, an information cannot be diversified (**S.2**) without existing in the dataset at the first place (**S.1**). Hence, it is possible to drop the looser specification **S.1** while maintaining the completeness of the requirement sheet. Since the opposite is not necessarily true i.e., an information might exist without being diversified, we keep both specifications to allow a more fine-grained comparison of existing datasets.

S3 : Contribution abundance

Besides the necessity of computing the similarity between tasks, being able to compute the performance of the workers from which the selection models are learned (See Section 3.3 in Chapter 3) is a key element of the learning, targeting and evaluation processes. For a given worker, this performance is computed using the correctness of his contributions for the tasks he participated in (See Section 3.3.2 in Chapter 3). A profile belonging to a worker who has not contributed to any task, or to a worker who randomly answered all the tasks he participated in, is not exploitable. Similarly, tasks for which no contributions, or random contributions [1] are collected are useless. We summarize the latter by the following specifications:

S 3.1 The dataset must contain a large number of contributions[a]. That is, both the tasks and the workers present in the dataset must have a reasonable number of contributions.

S 3.2 The dataset must contain non-random contributions for tasks and for workers. We show later how this can be achieved during the campaign design and the data preprocessing steps.

[a] At this point, the quantitative requirements in terms of number of contributions per worker and per task are not set. These are discussed later in this chapter.

In the next section, the evaluation datasets proposed in the literature are presented and their appropriateness with respect to Specifications **S.1**, **S.2** and **S.3** is assessed.

[1] This might occur with badly designed, poorly rewarded or subjective tasks. It is indeed less likely to occur when compared to random answers per worker. In fact, the randomness of the answers of a worker usually results from a malicious behavior.

4.2 Crowdsourcing evaluation datasets

4.2.1 An overview of the state-of-the-art

Table 4.1 details the characteristics of the evaluation datasets available in the crowdsourcing literature. For the sake of completeness, both publicly and privately available datasets are reported even though the latter ones are not usable for our evaluation. The table also shows the compliance of these datasets with Specifications **S.1**, **S.2** and **S.3** discussed earlier in this chapter. The compliance with these specifications is judged based on a set of observed characteristics in the dataset which we enumerate as follows :

- The worker features (*Feat.*) : this is a quantitative feature showing the number of worker profile features found in the dataset. This characteristic allows assessing the compliance of the dataset with **S.1.1** and **S.2.1**.
- The task content (*Cont.*) : this is a qualitative feature showing whether the dataset contains information about task content or not which allows assessing the compliance with **S.1.2**.
- The task diversity (*Div.*) : this is a qualitative feature showing whether the dataset contains more than one type of tasks or not. This reflects the conformity with **S.2.2**.
- The contribution density (*Den.*) : the set of contributions can be Dense (D), meaning that all of the tasks were solved by all of the workers, Semi-Dense (DS) meaning that the sets of workers who answered different tasks overlap or Sparse (S), meaning that the workers who answered one task are different from the workers who answered another task. This characteristic reflects the compliance of the dataset with Specification **S.3.1**

In the literature, many datasets have been used to evaluate crowdsourcing quality control techniques. However, only a few among them provide information about the declarative profile of the workers [53, 76] which is in line with the low number of quality control methods leveraging this aspect (See Section 2.2 in Chapter 2). The same observation was made by Ye et al. in [138] who highlighted the lack of worker related datasets in crowdsourcing. The previous reasoning also applies to the content of the tasks which is not always present in the datasets [53, 117]. On the opposite side, the contribution abundance requirement is almost met by all of the datasets [16, 53, 57, 68, 76, 117, 129]. This might be due to the fact that aggregation methods, which constitute a large part of the crowdsourcing related literature as shown in Chapter 2, usually require this kind of datasets.

The *Data For Everyone (DFE)*[2] corpus from Figure Eight provides a large number of real task sets for which many contributions have been collected. While these sets are varied enough in the task types, they suffer from at least one of the following limitations: First, the majority of them provide aggregated contributions instead of individual contributions, which violate Specification **S 2.1**. Second, to the best of our knowledge, none of these datasets provide the profiles of the workers which violates Specification **S 1.1**. Third, the content of the task is not always present which does not meet Specification **S 1.2**. One can argue that it is possible, through some data engineering effort [3], to combine a number of these sets

[2] `https://www.figure-eight.com/data-for-everyone/`

[3] Transferring missing data like profiles or individual contributions from one set to the other.

into a larger specification-fulfilling dataset. However, the datasets found in the DFE corpus are designed and generated independently by different requesters. Hence, the intersection between the workers and tasks of different datasets, when computable[4], might be empty or sparse which hinders any "match and transfer" step.

All of the aforementioned datasets are all *real crowdsourcing datasets*. That is, datasets generated through an actual crowdsourced data collection step. Nevertheless, *Synthetic datasets* have been also used in the literature. Roy et al. [107] and Rahman et al. [96] generated a set of workers and tasks distributed over a set of skills found in a multilayer skill taxonomy in order to test the efficiency of their skill matching approaches. Others, such as Welinder et al. [129] and Hung et al. [48] generated datasets to evaluate the performance of their aggregation algorithms. While generating synthetic evaluation datasets for aggregation and skill matching optimization approaches is relatively an easy and scientifically valid approach, generating synthetic datasets to evaluate approaches that leverage worker's behavior (e.g., fingerprinting [108]) and profile (e.g., declarative profile based worker selection [76]) is unfeasible. That is because, on one hand, ignoring the uncertainty and noise resulting from the subjectivity of the human being in generating the data, produces a dataset which does not reflect the real crowdsourcing context. And, on the other hand, modeling the uncertainty and noise is impossible due to the lack of behavioral study of the crowd in crowdsourcing systems. Hence, a synthetic dataset could, theoretically, fulfill all the specifications except Specification **S 3.2**.

The entries in Table 4.1 are filled as follows:

- **S1.1** is labeled "+" if at least one feature of the worker profile is available in the dataset (column Feat.). Otherwise it is labeled "-".

- **S1.2** is labeled "+" if the content of the tasks is available in the dataset (column Cont.). Otherwise it is labeled "-".

- **S2.1** is labeled "++" if worker profile features are present in the dataset and if the number of workers allows a sufficient representation of all the features (under the assumption of uniform distribution of the features over the observations). It is labeled "+" if he number of workers seems insufficient to cover the features. Otherwise it is labeled "-".

- **S2.2** is labeled "+" if different types of tasks are present in the dataset (column Div.). Otherwise it is labeled "-".

- **S3.1** is labeled "+" if both the tasks and the workers have sufficient contributors. Otherwise it is labeled "-". We consider 20 workers per task (which allows a good quality aggregation [75, 117]) and 20 tasks per workers (which allows to asses an accurate reliability of a worker) to be the thresholds for contributions abundance.

- **S3.1** is labeled "+" if the dataset is not synthetic. We consider that real datasets have been collected in a sound manner.

[4] e.g. for unaggregated or un-anonymized datasets.

To summarize, existing datasets as well as synthetic data generation are not satisfactory for our evaluation task. Therefore, creating a custom evaluation dataset that meets Specifications **S.1**, **S.2** and **S.3** is due. In the next section, and before jumping into the detailed explanation of the creation process of such a dataset, we extend this dataset-related discussion to point out the existence of an important gap in the crowdsourcing research community hindering a sound comparison of quality control methods. We call it "The Benchmarking Gap".

Table 4.1: A comparison of a sample of dataset used in the literature to evaluate crowdsourcing quality control.

Ref	Dataset	Characteristics									Compliance with our requirements					
		Worker		Tasks			Contributions		Pub.	RD	S1.1	S1.2	S2.1	S2.2	S3.1	S3.2
		#	Feat.	#	Cont.	Div.	#	Den.								
[53]	Stack overflow	505	8	14021	Yes	No	42063	S	-	+	+	+	+	-	+	+
	Evergreen webpage	434	9	7,336	Yes	No	22,008	S	-	+	+	+	+	-	+	+
	TREC 2011	160	9	1826	Yes	No	5478	S	-	+	+	+	-	-	+	+
[68]	Online product search	255	0	256	No	No	NA	S	-	+	-	-	-	-	NA	+
[75]	Synthetic	11	0	300	No	No	3300	D	-	-	-	-	-	-	-	-
[76]	Knowledge dataset	100	5	75	Yes	No	7500	D	-	+	+	+	-	-	-	+
	RTE	NA	5	80	Yes	No	NA	D	-	+	+	+	-	-	-	+
	Disambiguation data	277	5	50	Yes	No	13850	D	-	+	+	+	+	-	-	+
[117]	Affective text analysis	10	0	700	Yes	No	7000	D	+	+	-	+	-	-	-	+
	RTE	10	0	800	Yes	No	8000	D	+	+	-	+	-	-	-	+
	Word Similarity	10	0	30	Yes	No	3000	D	+	+	-	+	-	-	-	+
[129]	Image annotation Synth.	4-20 /task	0	500	No	No	NA	NA	-	-	-	-	-	-	-	-
	Image annotation Real	40 /task	0	100	No	No	4000	NA	-	+	-	-	-	-	+	+
[147]	Image labeling	109	0	807	No	No	NA	SD	-	+	-	-	-	-	-	+
	Relevance judgment	6 /task	0	2665	No	No	16000	S	-	+	-	-	-	-	-	+

Feat. : worker Features, **Cont.** : task Content, **Div.** : task Diversity, **Den.** : contribution Density
D : Dense contributions, **DS** : Semi-Dense contributions, **S** : Sparse contributions, **n/a** : not available
Pub. : Public availability, **RD**: Real Dataset, **-** : Un-fulfilled requirement, **+** : Fulfilled requirement

4.2.2 THE BENCHMARKING GAP

In fact, when judging the limitations of the existing datasets beyond the requirements of our specific evaluation objectives, from the preceding discussion emerges the existence of a wide gap in the crowdsourcing research. That is, besides the fact that barely a handful of the datasets used in the literature have been made public (See column *pub.* in Table 4.1), none of them provides sufficient data - at least qualitatively - in order to be used to evaluate and compare multiple approaches at a time. We explain this by the following example: suppose that we are interested in comparing two quality control approaches such as a skill-based task assignment method (see Section 2.2.3 in Chapter 2) and a profile-based worker selection method (see Section 2.2.4 in Chapter 2). A suitable dataset for evaluating the former method must contain a large number of contributions per worker for tasks of various types[5]. At the opposite side, to evaluate the latter method, a dataset containing the declarative profiles of the workers is needed. It is clear that none of the datasets described earlier contains both types of information to allow a sound comparison of the two methods. This reasoning can be extended to other approaches showing that benchmarking the existing quality control techniques is not possible due to the lack of appropriate evaluation dataset. Motivated by this challenge, we argue, in the next paragraph, that by adding one more specification to the previously set ones, we can take our dataset creation task one step further toward filling this benchmarking gap.

Table 4.2 details the requirements of a representative set of quality control methods in terms of the specifications of a suitable evaluation dataset. The majority of classical quality control methods such as aggregation techniques [14, 56] and qualification-test-based worker screening does not require any specific features to be present in the dataset aside from the set of contributions i.e. a set of labels indexed by an (ID_{worker}, ID_{task}) key. Other methods, such as profile-based and skill-based worker selection require the presence of the worker profiles[6] in the dataset. Methods which take into account the type of the task when selecting/screening workers - and which we refer to as contextual methods - necessitate either the existence of a category-labeled tasks or the content of the task from which the task type can be derived. Blue shades indicate a property covered by Specification **S.1**, **S.2** or **S.3** (from lighter to darker respectively) while red color indicates a property which is not covered by any of the three requirements. It is clear that a dataset that fulfills the specifications **S.1**, **S.2** and **S.3** meets - with few exceptions - the requirements of the majority of these methods. To cope with the remaining exceptions, two approaches are possible. One can either follow an ad-hoc approach, which consists in extending the existing specifications to cover the remaining scenarios, or one can adopt a more future proof approach by providing the tools to extend the existing dataset on-demand. We follow the latter approach whose requirements and specifications are explained as follows:

S4 : Extensibility

The creation of a generic and information rich dataset should always be open to new contributors, so that absent and new features can be proposed and collected based on uncovered and new quality control needs. Moreover, creating a realistic evaluation dataset for crowd-

[5] To allow computing the skill profile of each worker.

[6] Whether it is a declarative or a derived profile.

Table 4.2: The needs of selected quality control methods in terms of dataset content

Method	Workers	Task	Contrib.	Other
MV, WMV [56], EM [14]	ID	ID	Yes	n/a
Worker modeling [59]	ID	ID	Yes	n/a
Task modeling [131]	ID	ID	Yes	n/a
Qualification tests, test questions *	ID	ID	Yes	n/a
History-based *	ID	ID	Yes	n/a
Contextual history-based *	ID	Content	Yes	n/a
Programatic gold [91]	ID	ID/Content	Yes	Online interaction
Profile-based [61]	Profile	ID	Yes	n/a
Contextual profile-based [76]	Profile	Content	Yes	n/a
Skill-based [107]	Skill Profile	Skill Profile	Yes	n/a
Self-evaluation [53]	Profile	Content	Yes	n/a
Task composition[1]	Preferences	Content	Yes	n/a
Incentive based [77]	ID	ID	Yes	Reward variance

n/a: not available, * : Common methods in commercial crowdsourcing platforms.
Blue : requirement covered by Specifications **S.1** (light), **S.2** (Darker) and **S.3** (Darkest).
Red : requirement not covered by any of the specifications.

sourcing quality control necessarily passes by a crowdsourced data collection step, which is obviously a paid process. This makes the creation of a large enough dataset very costly, hence not achievable by only one entity (research laboratory, company, ...). Therefore, we add to the qualitative specifications **S.1**, **S.2** and **S.3** detailed earlier in this section a fourth specification as follows:

S4.1 The dataset must be collaboratively extensible both in terms of tasks, workers and contributions and in terms of worker features and task types.

In the remainder of this chapter, we show how we designed and built CrowdED to fulfill Specifications **S.1**, **S.2** and **S.3** and how CREX guarantees its extensibility to fulfill Specification **S.4**.

4.3 CrowdED : Crowdsourcing Evaluation Dataset

In this section, the process used to create CrowdED is described in details and the statistical and structural characteristics of the latter are presented. This process is divided into three steps shown in Figure 4.1: First, the data preparation step during which the raw resources such as the task input are collected and preprocessed. Second, the data collection step during which the actual contribution and profile crowdsourcing occurred. Finally, the data formatting step during which the collected contributions and profiles are cleaned and restructured.

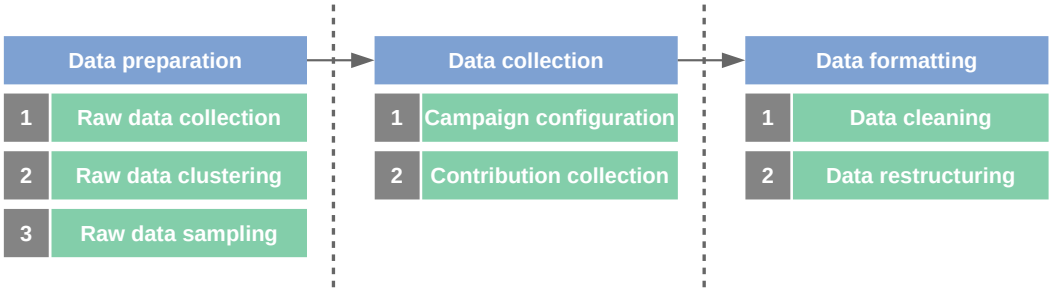


Figure 4.1: An overview of the creation process of CrowdED.

4.3.1 Data preparation

4.3.1.1 Raw data collection

We built our task corpus by collecting publicly available task sets from the Data For Everyone[7] (DFE) datasets provided freely by Figure Eight[8] (FE). The main motivation behind choosing the DFE datasets is to use tasks that have served a real world application. In fact, it is possible to generate random labeling and knowledge related tasks from scratch and to use them in the dataset generation process. However, these will not be as significant as real world tasks. Furthermore, DFE is a sustainable source[9] of task sets for future extension of CrowdED (Specification **S 4.1**). Our initial task pool consisted of 280K+ tasks, originally belonging to 11 different task sets. The task content was distributed over various domains such as sport, fashion, politics, economy, disaster relief etc. and over different action types like relevance judgment, image labeling, tweet categorization etc. (Specification **S 2.2**). Table 4.3 summarizes the characteristics of the task sets used to build the task corpus. It is clear that the tasks are unevenly distributed over the various task sets. For instance, set "A9" constitutes 67% of the entire corpus. In order to balance our task corpus we sampled 4000 tasks out of each set (i.e. the size of the smallest set).

Output of the step : After balancing the initial task corpus, we ended up with a corpus of 44k tasks evenly distributed over 11 sets. In the next steps, this corpus is clustered, sampled and published on a crowdsourcing platform for contribution collection.

4.3.1.2 Grouping and sampling the tasks

The ultimate goal of the workflow explained in this section is to produce a task corpus that can be submitted to be crowdsourced by a large number of workers (specifications **S 2.1** and **S3.1**). Indeed, submitting the whole task set requires a very large budget. For instance, if we intend to submit the entire 44k tasks set under a reasonable reward and redundancy configuration, the campaign would roughly cost 80,000$ US[10]. Thus, a sampling step that aims at reducing the number of tasks to be crowdsourced while maintaining the dataset specifications is needed. A straightforward approach would consist in uniformly sampling a given number of tasks (fixed by the budget constraint) out of each task set. Nevertheless,

[7] https://www.figure-eight.com/data-for-everyone/
[8] Formerly named CrowdFlower.
[9] Yet, it is not the only one since any other task corpus can be used.
[10] Which is not feasible since, for this work, we had a budget equal to a fraction of this sum

Table 4.3: On overview of the task sets used to build initial task corpus of CrowdED.

Task set name	# Tasks	# Questions	Question type	Domain
A8	18129	3	MCQ	Disaster relief
A9	189000	2	MCQ/FT	Sport
BI	10672	2	MCQ	Natural sciences
CH	5702	1	MCQ	Natural sciences
DE	15702	1	MCQ	Fashion
11 FO	4000	1	MCQ	Sport
GO	13872	3	MCQ	Politics
PO	5000	3	MCQ	Politics
SO	10976	1	MCQ	Disaster relief
SM	4000	2	MCQ/FT	Technology
US	5015	1	MCQ/FT	Economy

MCQ: Multiple Choice Questions.
FT : Free Text answer questions.

since the objective behind creating the dataset is to compare and analyze what are the best configurations for the clustering/vectorization configurations (motivated in Section 3.3.1 in Chapter 3), we argue next that this approach is not suitable for our study.

Recall : For convenience, we recall and summarize here the discussion found in Section 3.3.1 concerning the task grouping step as follows: vectorizing and clustering the tasks are key steps in the learning process of CAWS. Indeed, vectorizing a task can be achieved with different techniques and features which might have different impacts on the quality of the task grouping and, thus, on the learning process. The same applies on the clustering algorithm used to group the tasks in homogeneous categories. We are therefore interested in comparing the impact of different vectorizing/clustering algorithm combinations on the quality of the learning and targeting process.

First, when drawing a small number (relatively to the corpus size) of random samples equally across all the tasks in the task corpus, we might end up with a sample which falls in one cluster of a given clustering which prevents comparing it with other clusters in the same clustering. Second, a set randomly sampled from the task corpus, and due to the major size reduction, might not be partition-able by any or all of the clustering algorithms we are interested in comparing. In other words, clustering such a set might not converge or might not reflect the real behavior of a given clustering algorithm for the considered type of tasks.

Therefore, we propose below a sampling process which starts by grouping the tasks using the various vectorizing/clustering combinations to produce multiple clustering systems[11] of the same task corpus. Then, we apply a sampling algorithm that yields one sample task set which, on the one hand, respects our budget constraint and, on the other hand, is representative of all of the produced clusterings. It is worth noting that two-step sampling strategies are commonly used in mixed implementation studies i.e., studies that require qualitative and quantitative proofs. Our approach is inspired from these strategies of which different implementations and variants are explained by Palinkas et al. in [92].

[11] i.e. a clustering yielded by a combination of one clustering method and one vectorizing technique

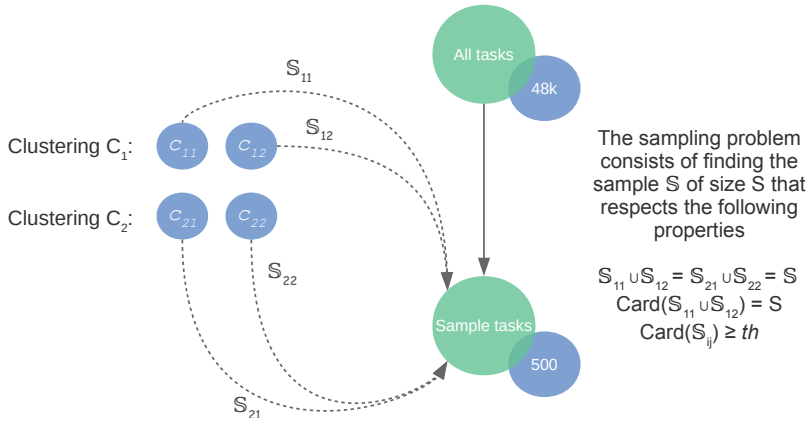


Figure 4.2: An instantiation of the constrained sampling problem. Here, two clusterings systems of two clusters each are considered. The maximum sample size S and the minimum sample-per-cluster size is th.

Vectorizing and clustering the tasks

For the reasons detailed in the previous chapter, we chose to compare the following vectorization techniques: TFIDF, TFIDF with PCA and Doc2vec. Moreover, we consider the following clustering algorithms which are representative of the three main existing clustering approaches i.e., partitional, density-based, and hierarchical : *Kmeans*, *DBSCAN* and *agglomerative clustering*. To prepare this comparison, we started by vectorizing the tasks in the task corpus using each of the three stated techniques and clustered them using one combination of clustering algorithm and feature vector at a time. We ended up with nine clustering systems of the same task corpus to be compared.

Indeed, in our case, the quality of each combination is judged by the quality of the learned model it produces which, in turn, is judged by the quality of the selection process it induces. In other words, the aim is not to optimize a given clustering quality measure such as the Silhouette [106] or the V-measure [104]. Yet, it is important that the used vectorizing/clustering combinations show a satisfactory validity and stability each. Therefore we conducted an extensive clustering study and measured the silhouette, homogeneity, completeness and V-measure of each clustering for a set of parameters for the vectorizing (e.g., TFIDF vector size, Doc2vec window and vector sizes, PCA vector size, etc.) and the clustering (e.g., number of clusters, distance metric, DBSCAN density parameters, Agglomerative algorithm linkage, etc.). As a result of this study, we ended up eliminating DBSCAN from the clustering algorithm list since it showed a high instability caused by its sensitivity to the density parameters. Similarly we dropped TFIDF with PCA from the vectorizing technique list since, aside from the computational benefit, did not impact the outcome of the clustering process when compared to TFIDF without dimension reduction.

Sampling the tasks

This step aims at producing a sample task set that is sufficiently representative of the initial task population and of the various clustering systems resulting from the previous step. In other words, for any cluster c of any clustering system, one must find in the final sample a sufficient number of tasks belonging to c so that this cluster could be studied during the

evaluation process (i.e. to be able to compute a profile model for it). Figure 4.2 illustrates our objective in a small example. Suppose that we have two clustering systems C_1 and C_2, each of which contains 2 clusters i.e., clusters c_{11} and c_{12} for C_1 and clusters c_{21} and c_{22} for C_2. We aim at finding one representative sample $\mathbb{S}$ of a preset size S (defined by the budget). For $\mathbb{S}$ to be representative of both clustering systems, one should be able to find, in $\mathbb{S}$, at least a number th of tasks belonging to each of the clusters c_{11}, c_{12}, c_{21} and c_{22}. Solving the problem in the case of one clustering system can be straightforwardly achieved through stratified sampling [88] which consists in dividing the corpus into sub groups (which is already done by the clustering) and then to randomly select the task samples proportionally from the different subgroups or to randomly sample th points from each subgroup i.e., cluster. Nevertheless, solving it in the case of multiple clustering systems is more complicated. Here, classical stratified sampling is not possible since the subgroups i.e., the set of all the clusters that divide the population might, and most likely do, overlap. In other words, in the previous example, while the clusters belonging to the same clustering system are disjoint because we are using deterministic clustering algorithms (e.g., c_{11} and c_{12} are disjoint), clusters belonging to different clustering systems are not disjoint and might contain common tasks (e.g., c_{11} and c_{21} might overlap). By randomly sampling th tasks from each of the 4 clusters we might be selecting the same task twice. The problem becomes more complicated as the number of clustering systems and of clusters in each of them get bigger. Therefore, we propose to constrain the random sampling step to overcome the overlapping problem. We formalize this challenge as follows:

Problem input :

Let I, J, T be the number of clustering systems, of clusters[a] and of tasks respectively.
Let th be the minimum number of sampled tasks per cluster.
Let S be the total number of tasks in the sample.
Let Y be an $(I \times J, T)$ occurrence matrix such that:

$$Y_{ijt} = \begin{cases} 1 & \text{if } t \text{ belongs to the cluster } j \text{ in the system } i \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

[a] To simplify, we assume that all of the clustering systems yield the same number of clusters.

Problem variables :

Let t be a task in the corpus, j a cluster and i a clustering system.
Let $X_{ijt} \in \{0,1\}$ be the selection decision such that:

$$X_{ijt} = \begin{cases} 1 & \text{if } t \text{ is selected from } j \text{ in } i \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

Problem objective : We want to minimize the following objective function :

$$f_{(X)} = \frac{1}{I \times J} \sum_{i=1}^{I} \sum_{j=1}^{J} ((\sum_{t=1}^{T} X_{ijt}) - th)^2 \tag{4.3}$$

This is the mean squared error (MSE) of the cluster sizes w.r.t the optimal cluster size i.e., th. The intuition behind using MSE is that with sample size constraint (S) it yields perfectly balanced clusters at its minimum value. Having balanced size clusters, is the ideal but not mandatory, scenario to compare, on a fair basis, the learning performed over different clusters.

Let s_{ij} be the size of a cluster j in a system i, that is :

$$s_{ij} = \sum_{t=1}^{T} X_{ijt} \tag{4.4}$$

The sample size is constrained to be equal to S for each system i, that is :

$$\sum_{j=1}^{J} s_{ij} = S \tag{4.5}$$

Then total MSE of the system can be expressed as:

$$MSE = \frac{1}{I \times J} \sum_{i=1}^{I} \sum_{j=1}^{J} (s_{ij} - th)^2 \tag{4.6}$$

$$= \frac{1}{I} \sum_{i=1}^{I} \frac{1}{J} \sum_{j=1}^{J} (s_{ij} - th)^2 \tag{4.7}$$

$$= \frac{1}{I} \sum_{i=1}^{I} MSE_i \tag{4.8}$$

The value of MSE is minimal when all the MSE_i are minimal since they are positive independent values. We now compute the minimum of an MSE_i. Thus, we consider one clustering system of J clusters.

$$\sum_{j=1}^{J} s_j = S \tag{4.9}$$

$$\sum_{j=1}^{J} (s_j - th) = S - J \times th \tag{4.10}$$

$$\left(\sum_{j=1}^{J} (s_j - th) \right)^2 = (S - J \times th)^2 \tag{4.11}$$

Using the Cauchy-Schwarz inequality for sums :

$$\left(\sum_{j=1}^{J}(s_j - th)\right)^2 = \left(\sum_{j=1}^{J} 1 \times (s_j - th)\right)^2 \leq \sum_{j=1}^{J}(1)^2 \times \sum_{j=1}^{J}(s_j - th)^2 \quad (4.12)$$

$$\left(\sum_{j=1}^{J}(s_j - th)\right)^2 \leq J \times \sum_{j=1}^{J}(s_j - th)^2 \quad (4.13)$$

Hence :

$$(S - J \times th)^2 \leq J \sum_{j=1}^{J}(s_j - th)^2 \quad (4.14)$$

$$\left(\frac{S}{J} - th\right)^2 \leq \frac{1}{J}\sum_{j=1}^{J}(s_j - th)^2 \quad (4.15)$$

$$\frac{1}{J}\sum_{j=1}^{J}\left(\frac{S}{J} - th\right)^2 \leq \frac{1}{J}\sum_{j=1}^{J}(s_j - th)^2 \quad (4.16)$$

The right part of the equation is the MSE of a given distribution of s_i and the left part is the MSE of a distribution where all s_j are equal to $\frac{S}{J}$. Accordingly, the lower bound of the MSE, when the size constraint S is imposed, is met if and only if[12] all points are distributed at a fixed distance from the cluster size constraint th.

Problem constraints : Table 4.4 shows the constraint formalization of this problem. We explain them as follows:

- C_1 each selected task in a system i should be selected in all systems.
- C_2 The minimum number of selected tasks in each cluster should be greater than or equal to a threshold th.
- C_3 The total size of the sample is S.
- C_4 Trivially, one cannot select a task t from a cluster j in system i unless it is in this cluster.

Table 4.4: The sampling problem constraints.

Constraints	C_1	$\prod_{i=1}^{I}(1 - \sum_{j=1}^{J} X_{ijt}) + \prod_{i=1}^{I}\sum_{j=1}^{J} X_{ijt} = 1$	$\forall t = 1...T$
	C_2	$\sum_{t=1}^{T} X_{ijt} \geq th$	$\forall i = 1...I, \forall j = 1...J$
	C_3	$\sum_{j=1}^{J}\sum_{t=1}^{T} X_{ijt} = S$	$\forall i = 1...I$
	C_4	$X_{ijt} \leq Y_{ijt}$	$\forall i = 1...I, \forall j = 1...J, \forall t = 1...T$

Solving the problem We try to minimize $f_{(X)}$ subject to the constraints C_{1-4}. This is a complex[13] optimization problem which we propose to address using a random search

[12] The Cauchy-Schwarz inequality turns into an equality when the sequences in the sums are proportional. That is in our case, the 1s and the s_j.

[13] We are currently working with mathematicians to reduce this problem to a known form of quadratic optimization.

algorithm [97, 141] and to pick the first solution (i.e., task sample) that respects our constraints with a certain tolerance. Allowing a slight tolerance over the sample size can drastically reduce the time of sampling convergence while marginally affecting the requester budget (which is the origin of the sample size choice).

Figure 4.3 depicts the iterative process of the sampling algorithm and Algorithm 3 shows its detailed steps. The sampling starts by uniformly picking a sample *sol* of size S from the task population (Υ) (*Line 2* in Algorithm 3), the population is updated by removing *sol* form it (*Line 3*) and the initial fitness (MSE) is computed (*Line 4*). Then, until either all the constraints or the maximum number of iterations is met (*Line 6*), the following iterative process occurs. A new sample sol_{new} is picked in the neighborhood of the current sample *sol* by substituting λ tasks from it by new tasks randomly picked from the current population (*Line 7*). The new fitness is computed (*Line 8*) and, if it is improved, the new sample replaces the current solution. Otherwise, a new sample in the neighborhood of *sol* is chosen. The advantage of MSE is its sensitivity to high errors which means that very big and very small clusters will most likely be penalized after each positive iteration which helps a faster convergence of the algorithm.

Parameter selection The budget which we could dedicate to the data collection step is ruled by three variables. First the number of tasks i.e. S, second the number of workers and third, the reward to be paid per assignment. The budget is the product of all three of these variables. One must find the trade off between these variables in order to respect the budget while allowing a sound evaluation of CAWS using the collected data. On one side, a very low number of workers would render the learning process impossible as this induces a low number of observations per feature and thus can lead to an overfitting problem (Equation 3.6). On the other side, a very low number of tasks penalizes either the genericity of the corpus w.r.t the task types or the number of clusters that we would be able to create. Finally, a very low reward can imply a very long data collection time and a high number of malicious contributions (due to worker frustration). After tunning the values of the parameters we set S to 525, the number of workers to 450 and the reward (including the bonus) to 3 US dollars for completing all the tasks (assuming that all the workers will be eligible to get a bonus). The tasks in each cluster are meant to be split in a training set which will be used to compute the worker accuracies for tasks in the history and a testing set which will be used to estimate the worker accuracy when testing the targeting phase. In Figure Eight, quiz mode uses 1 to 20 tasks to estimate the reliability of the workers [27]. Li et al. [76] used 5 tasks in a probing stage to compute their reliability. Consequently, we set th to 30, which allows to split each cluster to a significant size learning and testing datasets. With th set to 30, the theoretical upper bound of the number J of clusters was equal to 17 ($th \times J \leq S$).

A discussion on the random sampling process In fact, there are no guarantees of convergence for the sampling algorithm we propose. In our case, running it with the aforementioned configuration, often returned a constraint-respecting sample in reasonable time. This can be the result of the initial clusters not being extremely unbalanced. Very unbalanced clustering might lead some clusters to be very small. In which case,

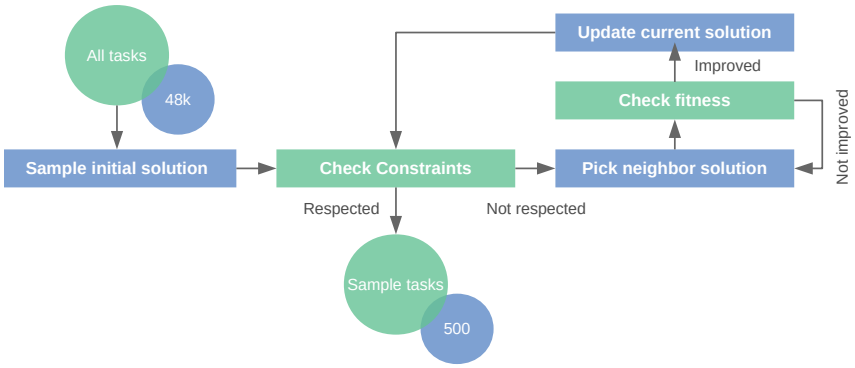


Figure 4.3: An overview of the iterative process of the random search.

the initial random sample (i.e., *line 2* in algorithm 3) can contain very few, or no, points belonging to this cluster. Thus, sampling might take too long to converge. To avoid this, one can initialize the algorithm differently. For instance, in each clustering system, a balanced sampling can be performed over all the clusters by selecting $\frac{S'}{J}$ tasks in each cluster, where J is the number of clusters in each system and $S' = \frac{S}{J}$. In this case, duplicate selection must be accounted which might reduce the size of the sample. To deal with this, one can complement the sample with the needed number of tasks by random sampling. By doing so, one is guaranteed to start with balanced initial sample.

Output of the step : With the kept clustering algorithms (i.e., K-means and agglomerative) and vectorizing methods (i.e., TFIDF and Doc2vec) we generated 4 different clustering systems of 15 task clusters each. Then, we sampled these systems by running the sampling algorithm with $th = 30$ and $S = 525$ and ended with a set of 525 tasks (which respects our budget) that were formated and submitted to FE (Figure Eight) for contribution collection.

4.3.2 Data collection

Talking the sample set as input, we designed a crowdsourcing job and submitted it to FE. Workers who selected the job were asked to read a detailed description of the task solving process and conditions and to fill their contributor IDs. These who decided to proceed with the job completion were redirected to an external web page on which the data collection took place. In the first stage of the campaign, we asked workers to fill their contributor IDs again (for an easier matching and control) and to answer a set of profile related and self-evaluation questions (Specification **S 1.2**)(see Section 4.3.3). Once done, workers proceeded in the actual task solving. For each job instance, tasks were randomly distributed over 11 pages in order to prevent the concentration of the negative impact of weariness on one subset of tasks. After completing the whole task set, a unique submission code was provided to each worker allowing his to receive his reward on FE.

Workers were rewarded a base pay equal to 1$ US. Additionally, a bonus of 2$ US was awarded (manually) to workers whose answers and declared profiles were of a good quality and high consistency (see Profile Rating in Section 4.3.3). Moreover, we estimated the job

Algorithm 3: The sampling process

Input : Cl: set of clusters, S: sample size, th: sampling threshold, itt: maximum number of iterations, λ: neighborhood, Υ: task population

Output : sol: the task sample that respect our constraints

```
1  sample(Cl, S, th, itt, λ, Υ) :
2      sol ← Υ.Rand(S); // randomly select an initial sample of size S
           from Υ
3      Υ_new ← Υ.sub(sol); // remove the current sample from the population
4      f_old ← computeFitness(sol, th); // compute the fitness Eq. 4.3
5      curItt ← 0 // initialize the iteration counter
6      while !checkConstraint(sol, th, Cl) and curItt < itt do
7          sol_new ← findNeighbour(Υ_new, sol, λ); // find a neighbour solution
8          f_new ← computeFitness(sol_new, th); // compute new fitness Eq. 4.3
9          if f_new < f_old then
               // if fitness improved then consider the new solution
10             sol ← sol_new;
11             Υ_new ← Υ.sub(sol);
12         curItt++; // increment the itteration counter
13     return sol;
14 checkConstraint(sol, th, Cl) :
15     result = True;
16     foreach cl ∈ Cl do
17         result←(count(sol, cl) ≥ th) // count the number of sampled tasks
               for each cluster and check if th is respected
18         if !result then
19             Break;
20     return result;
21 findNeighbour(Υ, sol, λ) :
22     vec1 ← Υ.Rand(λ); // rand. select a new task set of size λ in Υ
23     vec2 ← sol.Rand(λ); // rand. select a task set of size λ in sol
24     sol_new ← sol.replace(vec2, vec1); // replace vec1 by vec2 in sol
25     return sol_new;
```

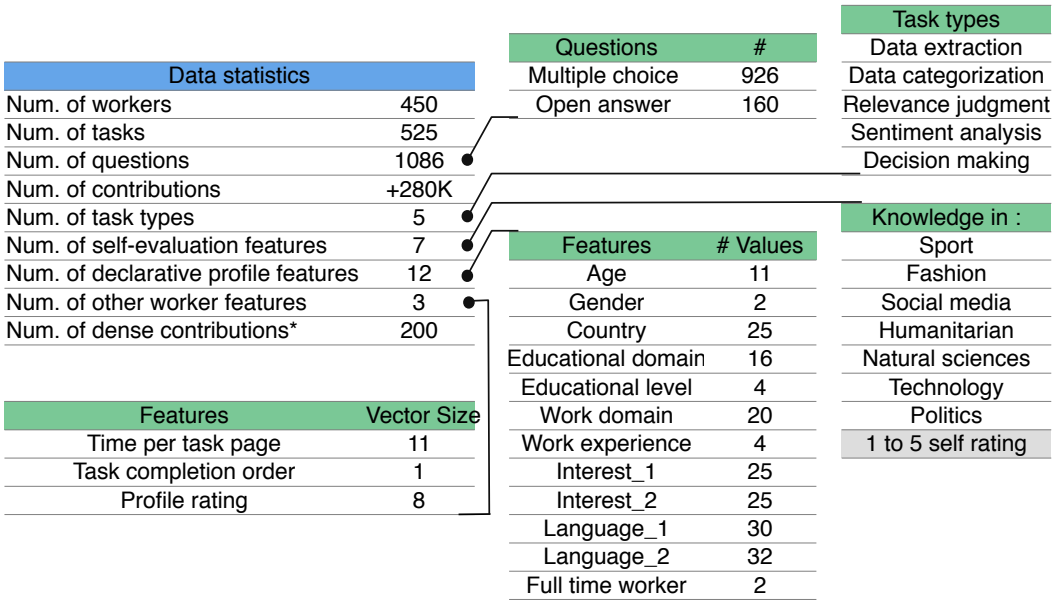


Figure 4.4: An overview of the structural characteristics of CrowdED. (*) a dense contribution is a set of answers given by a single worker to the entire task set.

completion time by 45 minutes, thus workers who finished the job in a very short time (i.e., less than 40 minutes) were automatically eliminated, with their contributions, and did not receive any reward. Finally, we only accepted workers of at least level 2 in the FE worker classification[14]. On one hand, these three parameters i.e., the bonus, the contribution duration and the minimum worker level, were strict enough to ensure that malicious workers (i.e., workers who intentionally fill random or wrong answers) are eliminated (Specification S 3.2). On the other hand, they are loose enough to allow a real representation of the quality issue in crowdsourcing. Contributions were collected during 3 months over all week days and covering all times of the day.

Output of the step : At the end of the data collection step we collected 2 types of data: first, the contributions of the worker for the tasks they completed. Second, the declarative, self-evaluation and behavioral profiles of these workers.

4.3.3 Data Structure and statistics

Figure 4.4 shows the structural characteristics of CrowdED as well as the features of tasks and workers that it contains. In total we collected 280K+ contributions for 525 tasks from 400 workers among which 200 completed the entire set of tasks. We call the set of contributions given by these 200 workers a "dense set". Structurally, CrowdED consists of 4 files: *contributions.csv* which contains the worker contributions, *workers.csv* which contains the worker profiles, *rating.csv* where profile ratings are stored and finally *task.zip* where the tasks content and description are stored. CrowdED has been made public, it is available on

[14]FE levels range from 1 to 3 where level 3 represents the most experienced and reliable workers and 1 represents all qualified workers

Figshare, on Github (See Appendix A) as well as on the project "Project-Crowd "page [15].

4.3.3.1 Tasks

Some of the 525 tasks in CrowdED contains up to 3 independent questions. Thus, the total number of answered questions is 1086. The majority of these questions (926) are multiple choice questions and the remaining part consists of open answer questions. The input of the tasks are tweets, images, scientific article quotes or news articles and headlines. Their action types fall into 5 categories: *data extraction, data categorization, relevance judgment, sentiment analysis, decision making*.

4.3.3.2 Workers

For each worker, we collected a profile consisting of 21 features divided into 3 types:

Declarative profile

We collected 12 features consisting of the following demographical, education and interest related information about the user : *age, gender, country, education domain, education level, work domain, work experience, interest 1, interest 2, native language, other spoken language and full time worker (i.e. whether the worker is a full time or occasional crowdsourcing worker)*. Figures 4.5, 4.6 and 4.7 report the distribution of the workers over the set of demographical, knowledge related and interest related features. We draw the following observations related to these distributions: these numbers are, for their majority, compliant with the numbers reported in previous studies found in the literature such as the study of the Mechanical Turk marketplace found in [52].

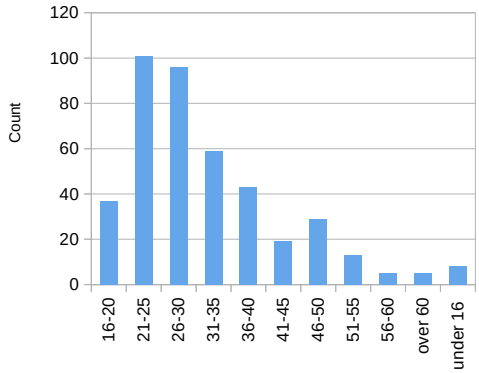
Self-evaluation features

We collected 7 features consisting of a 5-star self rating for 7 knowledge domains: *sport, fashion, technology, natural sciences, humanitarian work, politics, and social media*. Figure 4.8 shows the average self rating in various knowledge domains w.r.t. the gender of the worker. In average, female workers seemed more confident in their knowledge in fashion and humanitarian work, while male workers, rated themselves higher for sport. For the remaining domains, i.e., technology, natural sciences, politics and social media, both female and male worker rated themselves similarly.

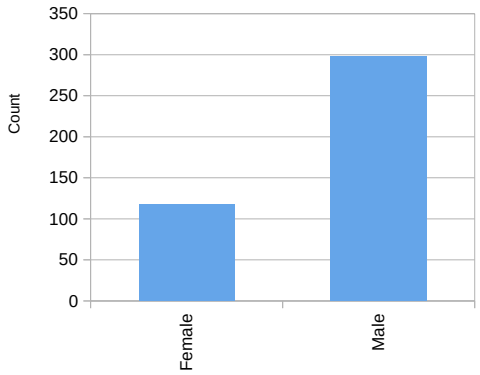
Behavior-related features

Four features related to the behavior of the worker during the campaign were collected. Three out of these features were collected automatically in the interface : *time for completing a task page, time for reading the description and filling the profile and the order of task completion*. The fourth, however, resulted from a complementary crowdsourcing campaign; in fact, in order to judge the consistency and reliability of the worker declarative and self-evaluation profiles, we ran a profile rating job on FE during which the profile of each worker who participated to our job was rated (from 1 to 4) for consistency by at least 7 workers. Figure 4.9 depicts the description of the task as published on Figure Eight.

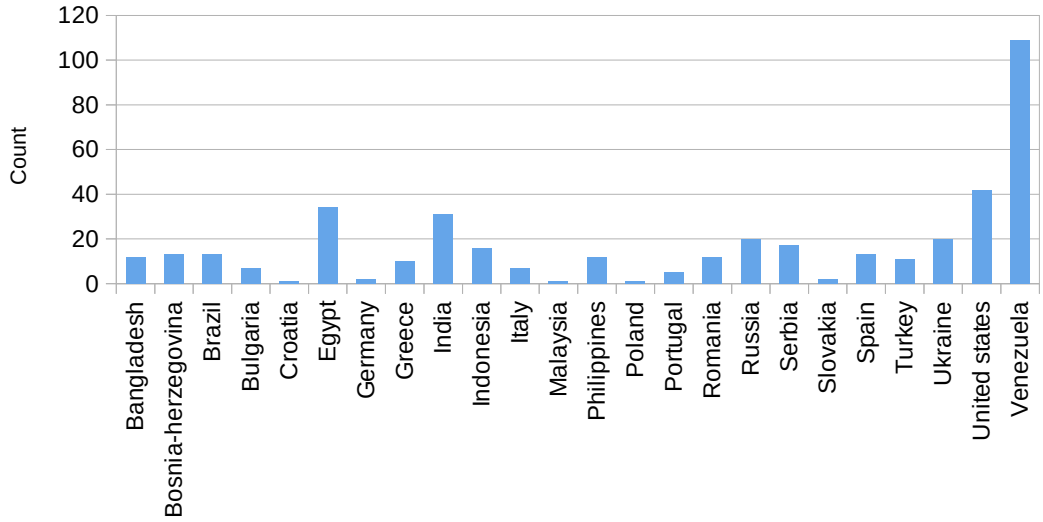
[15] `http://project-crowd.eu/crowded`



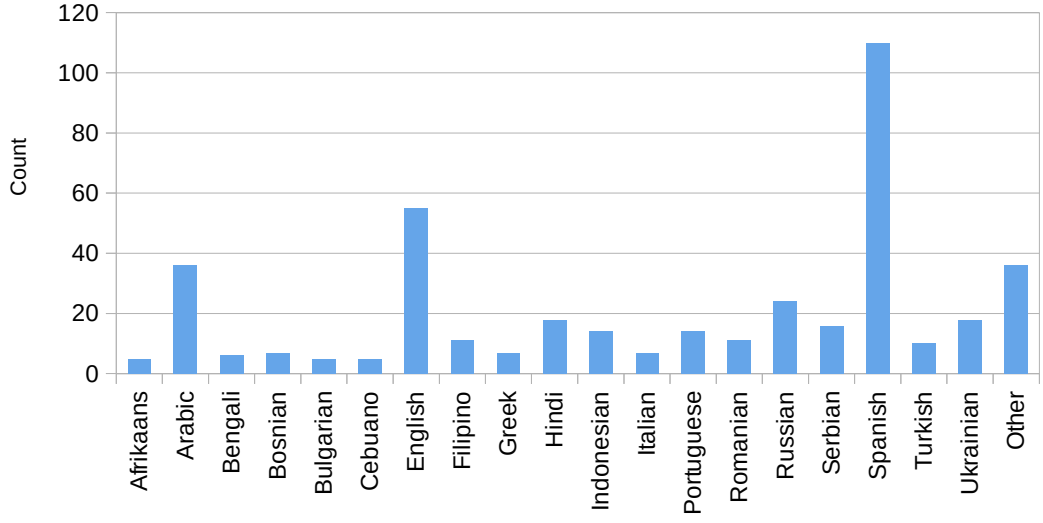
(a) Age



(b) Gender

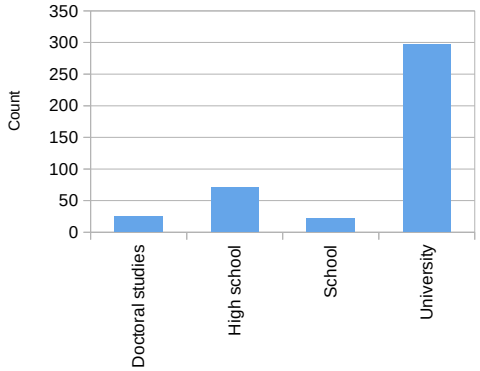


(c) Country

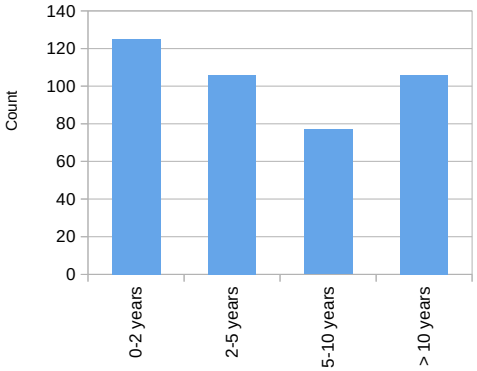


(d) Native language

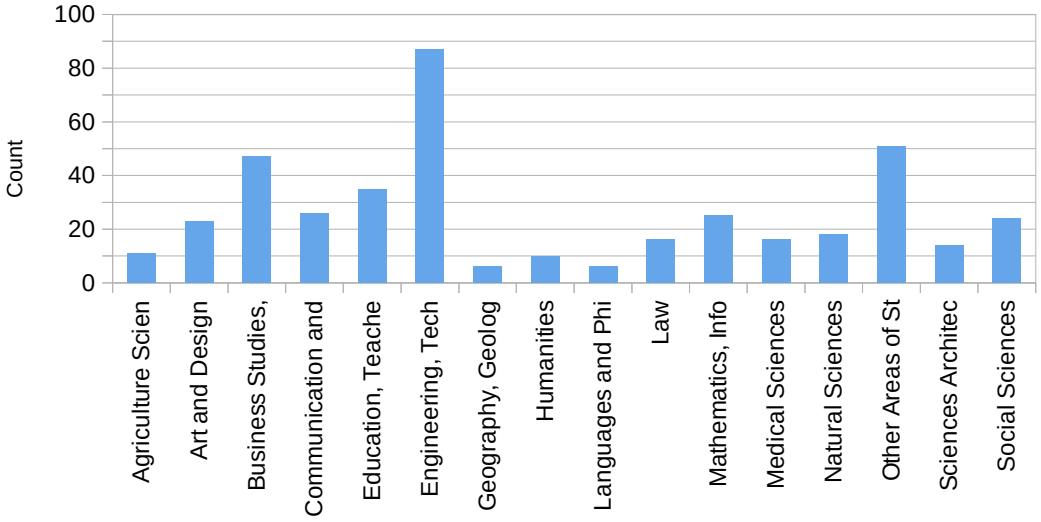
Figure 4.5: The distribution of workers over a set of demographical features: (a) age, (b) gender, (c)native language and (d) country.



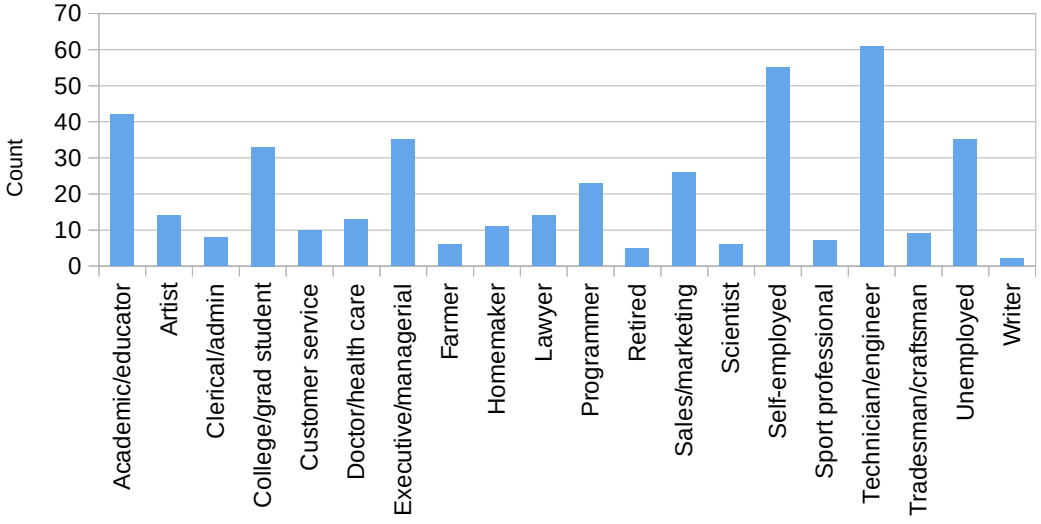
(a) Education level



(b) Work experience

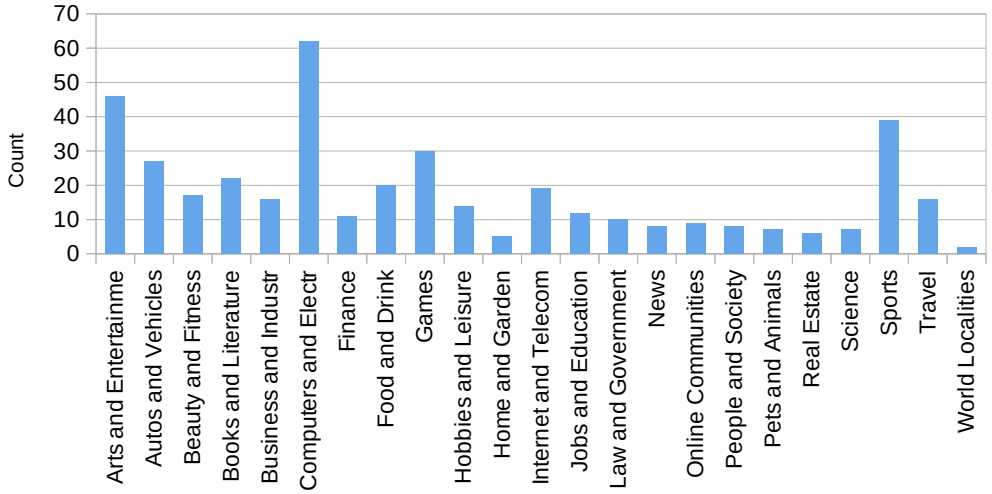


(c) Education domain

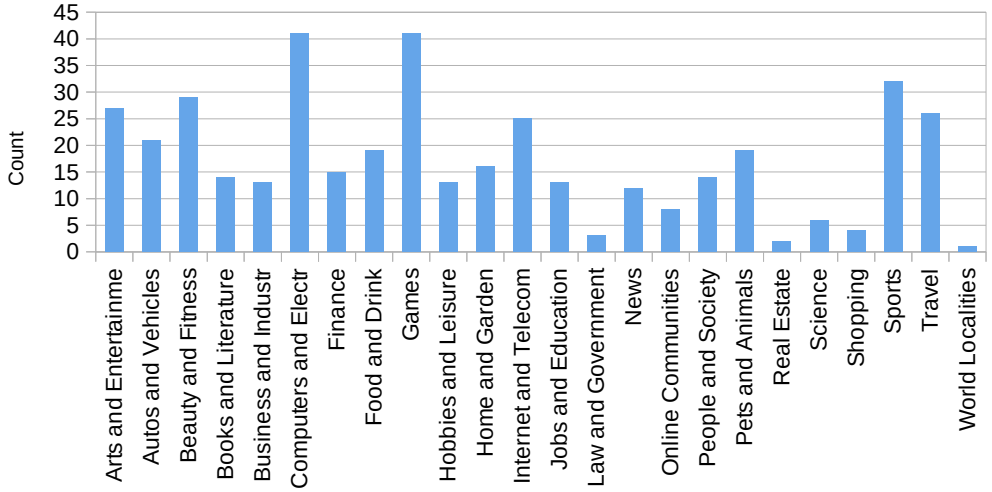


(d) Work domain

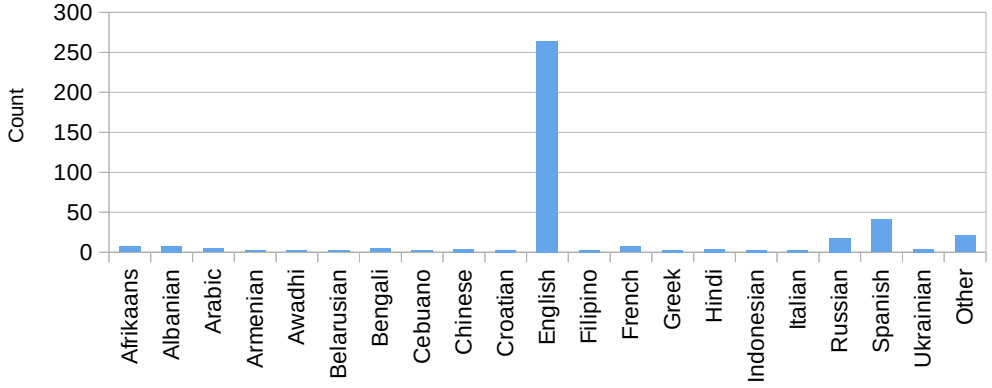
Figure 4.6: The distribution of workers over a set of knowledge related features: (a) education level, (b) work experience, (c) education domain and (d) work domain.



(a) Interest 1



(b) Interest 2



(c) Spoken language

Figure 4.7: The distribution of workers over a set of interest related features: (a) and (b) interests and (c) other spoken language.

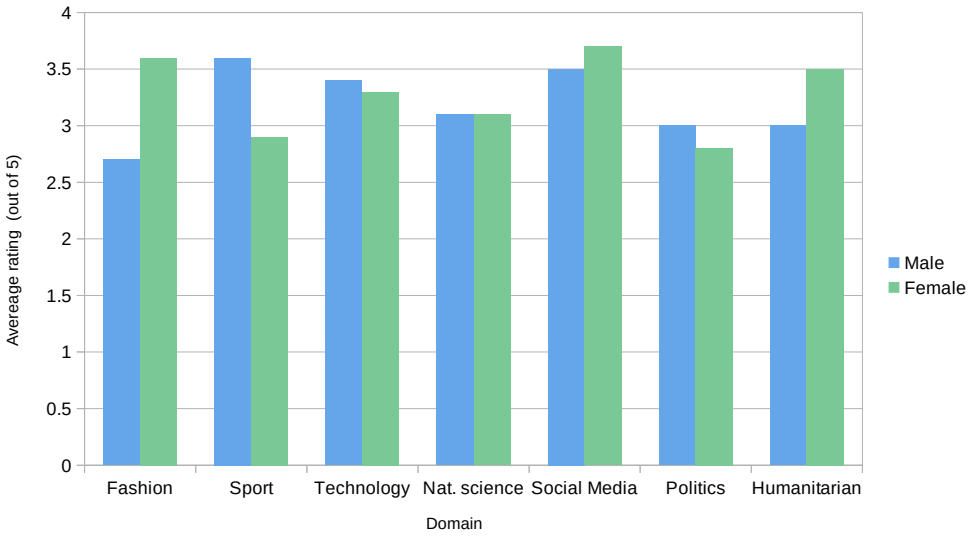


Figure 4.8: The distribution of the self evaluation rating for different knowledge domains w.r.t. the gender of the worker

4.4 CREX: CREATE ENRICH EXTEND

In this section, the CREX framework that helps extending and enriching CrowdED (or creating similar datasets) is described. As depicted in Figure 4.10, CREX uses a two-component architecture. The first component, CREX-D, allows a configurable task data selection while the second, CREX-C, provides tools to automatically generate crowdsourcing campaigns from the output of CREX-D. Figure 4.11 shows the possible parameters of each configurable module in CREX. The computational modules of CREX are developed with Python3. It uses well established and sustainable natural language processing and machine learning libraries such as *scikit-learn*[115], *nltk*[90], *gensim*[38] etc [16]. The web user interface uses a combination of *Bootstrap*, *JavaScript* and *PHP* and the used database technology is *MySQL*. A list of dependencies and of a description of the configurable parameters of CREX can be found in Appendix A. Moreover, a functional demo of CREX can be tested on the following URL `http://crex.project-crowd.eu/`.

4.4.1 DATA PREPARATION COMPONENT (CREX-D)

A typical crowdsourcing workflow consists of 3 steps: first, designing the task, second, crowdsourcing the task and last, collecting the results. Indeed, this typical workflow is suitable for classical crowdsourcing where the aim of the requester is to exploit the results in a limited application-centric way, e.g., label multimedia data to facilitate indexing, translate a given corpus, etc. In other words, it suits applications where the input data are fixed and limited in size. When it comes to research-related crowdsourcing, e.g., building evaluation, validation or training datasets where the usage of the collected data goes beyond the limited exploitation, the input data space is usually huge and more complex. Therefore, an upstream

[16] A full list of the CREX's dependencies can be found on `http://crex.project-crowd.eu/help.html`

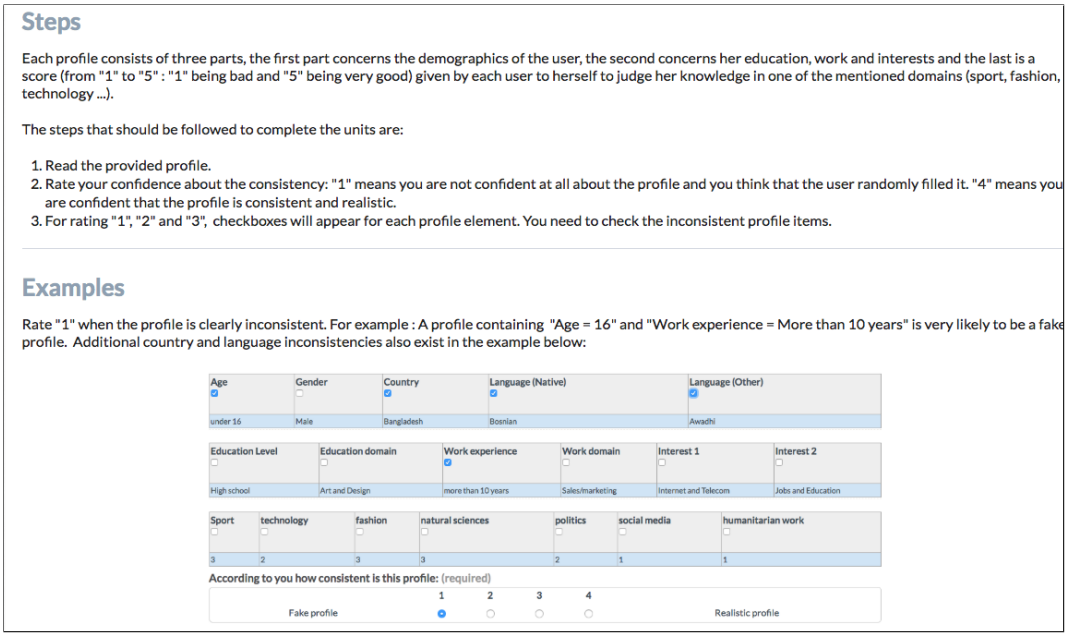


Figure 4.9: The description of the profile rating task.

input data selection effort is needed. A more suitable workflow is then a four step process that adds an input data selection step at the beginning of the aforementioned workflow.

Figure 4.11 depicts the structure of the CREX-D component. It comprises four modules: the *vectorizing module (VM)* , the *clustering module (CM)*, the *sampling module (SM)* as well as the *evaluation module (EM)*. These modules are available and **inter-operable** yet **independent**. That is, each module can be used separately or as an entry point for the remaining steps or substituted by another module of equivalent role. This allows a more flexible usage and thus a wider cross-domains utility of CREX.

The vectorizing module:

Grouping the tasks starts by extracting the features of interest from the raw data. In this work, we consider textual data where each data point is the textual representation of a task. Despite being limited to this type of data, CREX makes it easy to bypass this limitation by either feeding pre-vectorized data to the *CM* or by adding custom vectorizing functions to the *VM*. The actual implementation of *VM* supports frequency based text representation (TFIDF [110]) and semantic document representation (Doc2vec [70]).

The clustering module:

The *CM* allows to cluster the vectorized tasks using one of three types of clustering algorithms: partitional (K-means), density-based (DBSCAN), and hierarchical (Agglomerative). A user can use either a cosine or an Euclidean distance during the clustering process. However, the CM provides the possibility to feed the algorithm with a custom pre-computed similarity matrix [17].

[17] This option must be used carefully; some similarity measures are not significant when used with some clustering algorithm e.g. cosine distance and K-means [145]

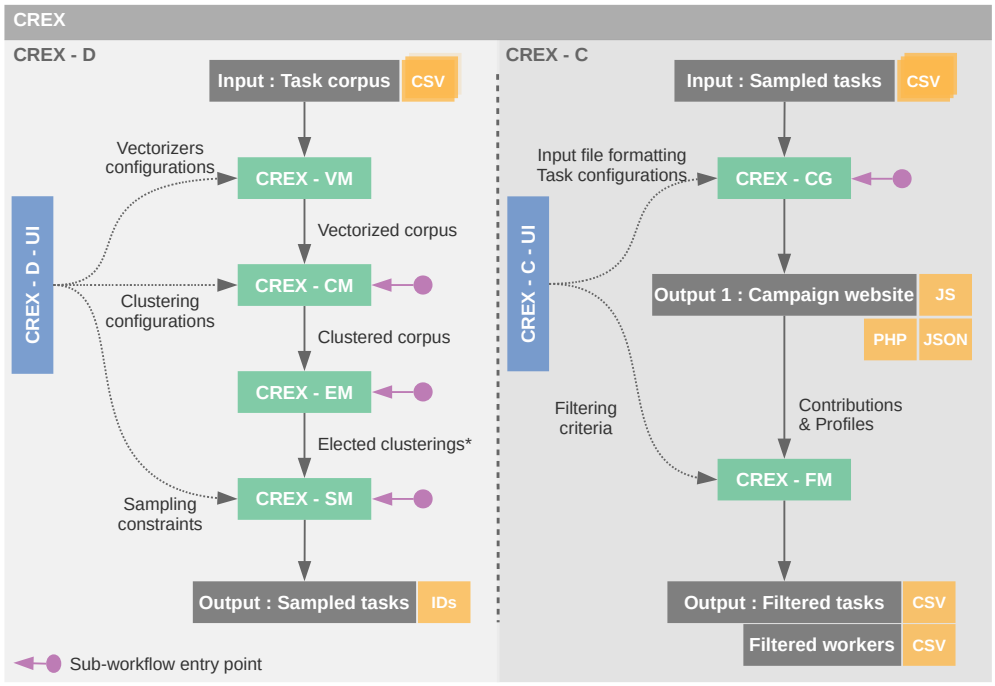


Figure 4.10: (An overview of the CREX framework that combines two main components: **CREX-D** for data selection and **CREX-C** for campaign generation and data collection.

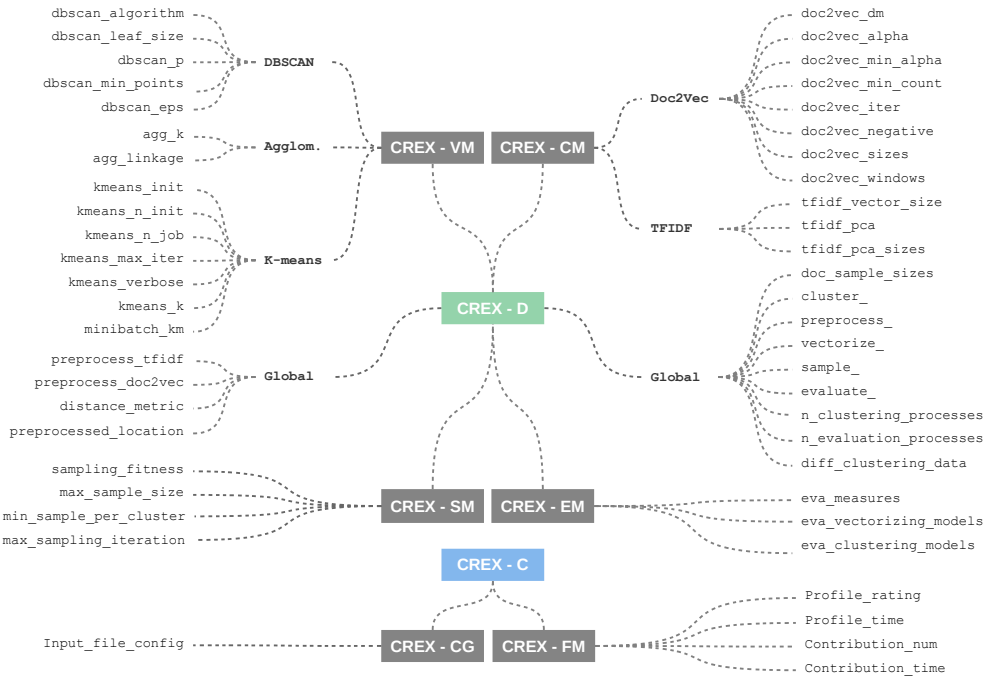


Figure 4.11: A summarized view of the CREX module configurable parameters.

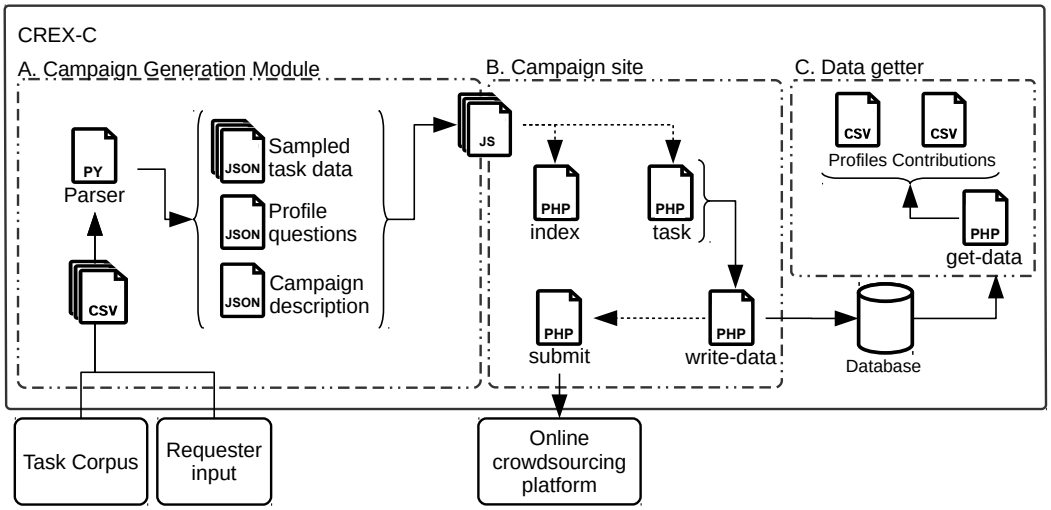


Figure 4.12: Overview of the CREX-C component.

The sampling module:

This module implements Algorithm 3 of the sampling process detailed in the previous section.

The evaluation module:

The *EM* module allows to evaluate the clustering process using internal and external validity measures such as silhouette [106], homogeneity, completeness and V-measure [104] as well as a custom validity measure consisting of a similarity to co-occurrence correlation matrix.

4.4.2 Campaign management component (CREX-C)

From a requester perspective, a mandatory step of the crowdsourcing workflow is the task design and generation. This step is a tedious and time consuming one due to two factors: first, the interest and use of crowdsourcing is growing to reach a wider sphere of scientific and social domains. Thus, the range of task forms and content is getting larger. Second, a crowdsourcing task, itself, might be dynamic i.e., it may require conditional or real-time computed components. Therefore, the capacity of commercial crowdsourcing platforms to provide practical design tools, preset templates and real-time computational means[18] is declining. A common way of dealing with these limitations is to build campaign sites with dedicated databases and back-end computational module and to make them accessible through a common crowdsourcing platform to provide reward payment and worker management. The campaign management component of CREX provides an easy-to-use tool for generating campaign sites from the sampled tasks (see Section 4.4.1) using the *campaign generator module* (CG). Figure 4.12 shows the structure of CREX-C, its interaction with the external libraries and the data flow in the system.

The campaign generation module:

CREX-CG takes 2 inputs. The first is the set of tasks to be published on the campaign site. The second is the requester input consisting of the task descriptions, examples and

[18] e.g., requester accessible back-end services or API to **dynamically** modify tasks and assignments.

instructions. It parses these inputs to intermediate JSON files and uses them to generate the campaign pages. The campaign site communicate directly with the database where the contributions and the worker profiles are stored. Contributions in the database are stored using a JSON format which allows a straightforward use of CREX-C for different task structures and types without the need to create a new database model and query rewriting.

The filtering module:

For a set of workers, tasks and contributions collected through the campaign generated by the GC module, the filtering module allows to select a subset of these data based on qualitative and quantitative selection criteria applied on the workers. These criteria cover the declarative profile features of the worker, their rating of their profiles, their time of task completion, their time of profile completion as well as the number of task they achieved. The code snippet 4.1 show an example configuration of the filtering module. In this example, we are interested in selecting workers, and their contributions, who completed at least 5 task pages (*line 4*), whose median profile rating is greater than 3 (*line 7*) and who took at least 1 minute to complete the profiles questionnaire (*line 10*).

```
1   # --- filter workers by criteria
2   criteria = {
3       # --- filter workers by number of completed pages
4       'worker_time_cont.csv': {'count': ['>=', 5]},
5
6       # --- filter workers by median profile rating
7       'Workers_decl_r_p.csv': {'median': ['>=', 3]},
8
9       # --- filter workers by time to complete the profile
10      'Workers_time_p.csv': {'Time_to_complete': ['>=', 60]}
11  }
12
13  # --- filter workers
14  filtered = select_workers_by_criteria(in_file, criteria, out_file)
```

Listing 4.1: Code snippet example showing the filtering module configuration

The filtering process has two main goals: First, it helps selecting a subset of the workers based on qualitative criteria to allow studying its characteristics e.g., its average performance of female workers. Second, and mostly, it allows to clean the data based on behavioral criteria. For instance, as shown in Figure 4.13 which reports the maximum rating per profile w.r.t the time spent by the worker filling it, a profile filled in less than 20 second is most likely to be inconsistent (maximum rating of 2^{19}). That is, it has been most likely filled randomly which means that the worker associated to this profile is very likely a malicious worker. Consequently, considering only the contribution of workers who spent a reasonable time answering the profile questionnaire would yields a noiseless dataset.

[19] Equivalent to "Profile with clear inconsistencies in our rating system"

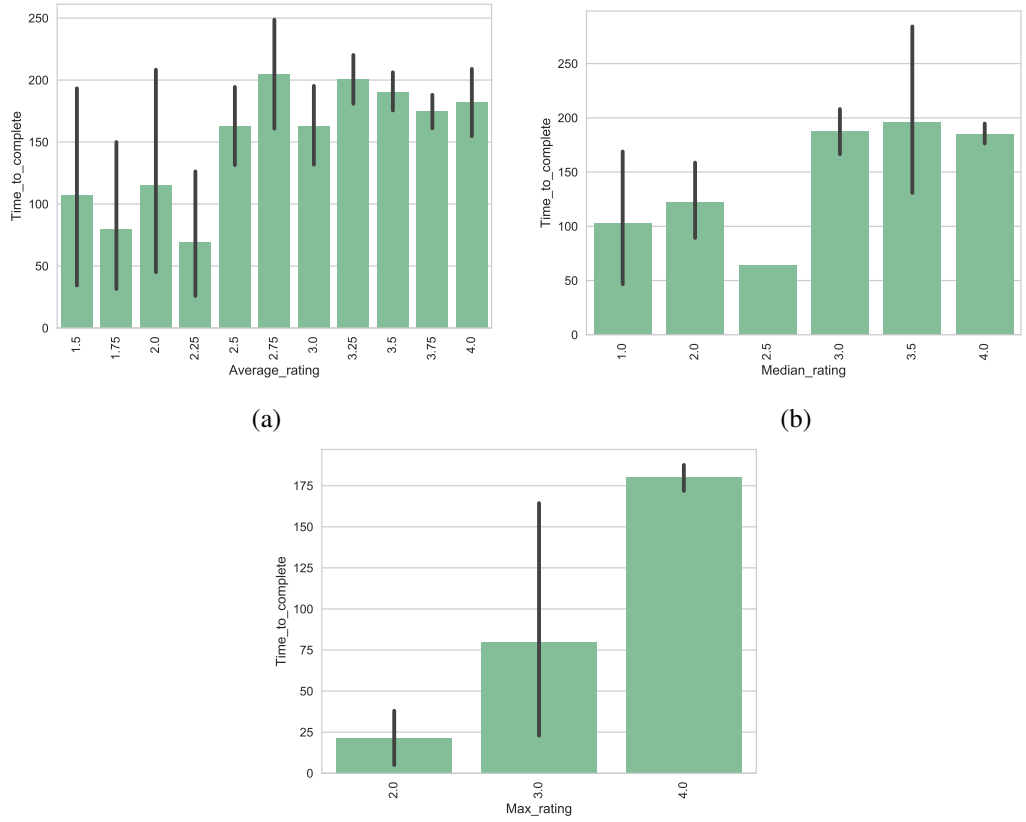


Figure 4.13: The relation between the quality of the profile (in terms of consistency) and the time spend to fill it : (a) Average rating, (b) Median rating and (c) Maximum rating

4.5 CrowdED and CREX re-usability

4.5.1 Usability in quality control evaluation

Table 4.5 shows the usability of CrowdED for evaluating the QC methods reported in Table 4.2. This usability is judged based on the needs of these methods in terms of information about workers, tasks and contributions and their availability in CrowdED. The majority of the methods that require information about workers and tasks only (regardless the type of the information) are supported by CrowdED. Others are supported either through *simulation*, i.e., vertically or horizontally splitting the dataset to simulate a real world situation like worker screening or through *augmentation*, i.e., adding more knowledge to the available data without the need for additional crowdsourcing by extracting new features or using external taxonomy to represent tasks and workers. Less frequent methods that require more information are not supported. Nevertheless, thanks to CREX (detailed in Section 4.4) they could be supported by extending CrowdED with a minor reconfiguration effort (e.g. changing the reward) or with a more demanding coding effort.

4.5.2 Compliance with the FAIR principals

The previous paragraph shows that CrowdED and CREX bridge, theoretically, the benchmarking gap in crowdsourcing. Yet, while fulfilling the theoretical specifications is crucial

Table 4.5: The usability of CrowdED w.r.t the existing quality control methods

	CrowdED support	
Method	Native	Extensible
MV, WMV [56], EM [14]	+++	n/a
Worker modeling [59]	+++	n/a
Task modeling [131]	+++	n/a
Qualification tests, test questions.	+++ S	n/a
History-based	+++ S	n/a
Contextual history-based	+++ S	n/a
Programatic gold [91]	- -	+ C
Profile-based [61]	+++	n/a
Contextual profile-based [76]	+++	n/a
Skill-based [107]	++ A	+++ Cf
Self-evaluation [53]	++	+++ Cf
Task composition[1]	- -	++ C
Incentive based [77]	- -	+++ C

A: supported through Augmentation, C: extensible through Coding, Cf: extensible through Configuration, S : supported through Simulation, n/a: not available, - : bad, + : good.

for the utility of the resources, these specifications cannot guarantee alone the re-usability of these resources which depends on their ease of use and their interoperability with existing tools. To guarantee this re-usability by the wide community (which allows a better extension and enrichment of CrowdED), the FAIR principles [135] (*Findable, Accessible, Interoperable, Reusable*) were considered during the design, the creation and the publishing process: CrowdED and CREX are available on Github and Figshare (with an associated DOI) which makes them **F**indable. They are published under CC and GPL licensing respectively to allow their **R**e-usablility and **A**ccessibility. CrowdED data are stored in *csv* files and no proprietary languages were used to develop CREX. This ensures the **I**nteroperability of the resources.

4.6 Summary

In this chapter we presented CrowdED, an information rich evaluation dataset. The first motivation behind creating CrowdED is to provide qualitatively and quantitatively sufficient data to evaluate multiple aspect of CAWS, i.e., the time and cost overheads, the selection quality and the grouping impact. This evaluation is not achievable through the existing evaluation datasets which lack for one or many of the data needed by the learning and targeting processes of CAWS i.e., worker profiles, task content and the contributions. We showed this in Section 4.2 through a comparative review of the literature datasets. To cope with this problem, CrowdED fulfills three specifications that we summarize as follows: (i) **Data richness** (**S1**) CrowdED is rich in information about the workers' profiles and the tasks' content. This has been achieved, first, by collecting three types of worker profiles i.e.,

declarative, self-evaluation and behavioral profile. (ii) **Data diversity** (**S2**) CrowdED reflects the diversity of tasks and worker profiles in a real crowdsourcing system. We completed this by collecting contributions for a task corpus containing tasks of various types e.g. relevance judgment, information extraction, etc. and belonging to different knowledge domains e.g. politics, sport, fashion, etc. Moreover, the distribution of the collected worker profiles is conform with the existing demographical studies found in the literature. (iii) **Contribution abundance** (**S3**) CrowdED contains over 280K contributions. All the tasks and the workers have a large number of contributions each.

Since crowdsourcing a large corpus is not possible due to the budget constraints, a sampling step of the initial corpus was mandatory. However, to enable evaluating the impact of the task grouping on the performance of CAWS as presented in Chapter 3, it is crucial to ensure that the sampled task set is representative of all the clustering systems produced by the vectorization and clustering algorithms of interest. Therefore, we proposed a two steps workflow: First, the initial (large) task corpus is clustered using the vectorization/clustering combinations to be compared. Second a constrained sampling algorithm is applied to find the set of tasks which, on one hand, is of a reasonable size to be crowdsourced and, on the other hand, is representative of all the clustering systems.

The second challenge motivating the creation of CrowdED is bridging the benchmarking gap found in the literature. We showed in Section 4.2.2 that none of the existing datasets allows a sound comparison of the quality control approaches which leverage different aspects of the crowdsourcing systems. The specifications **S1**, **S2** and **S3** fulfilled by CrowdED allow it to be usable in evaluating and comparing a wide range of existing quality control methods. In order to deal with the remaining exceptions i.e., the methods which are not natively supported by CrowdED, and to future-proof it, we proposed CREX. CREX is an open-source software framework that allows the extension of CrowdED to fulfill new qualitative e.g., new worker profile types, and quantitative requirements e.g., more contributions for a given task (**S4**).

In summary, this chapter introduced and described a novel information rich dataset for evaluating quality control in crowdsourcing and a new platform for collaborative creation of crowdsourcing evaluation dataset. The chapter also shed light on the challenges facing the creation of the dataset and the platform and on the solution proposed to cope with these challenges. Last but not least, it prepared the ground for a sound evaluation of our worker selection approach called CAWS. In the next chapter, the experiment setup and results of the evaluation process are detailed.

5

Evaluating the Context Aware Worker Selection Method

In this chapter, we evaluate the performance of CAWS in terms of quality. We focus on two main points: first, we evaluate the quality gain that CAWS can yield. This is done by varying the parameters of the system such as the clustering algorithms, the vectorizing algorithm, the learning model, etc. Second, we study the time and cost reduction that can be achieved by CAWS when compared to other worker selection methods.

Roadmap. In Section 5.1, we start by defining the evaluation measures that we will be using in the experiments. In Section 5.2, the general experimental setup of this evaluation is detailed. Then, in Section 5.3, the experiments, their objectives and their results are reported and discussed. Finally, we conclude this chapter in Section 5.4.

5.1 Evaluation metrics

Optimizing the quality and reducing the time and cost overheads are the two main goals of the method described in this work. In this section, we describe the metrics that we use to assess the ability of our method to achieve these goals.

5.1.1 Quality metrics

The final step of the crowdsourcing workflow is to aggregate the contributions collected from the workers to whom the tasks were assigned. This makes the individual accuracy of the workers less significant as the their contributions are transparent for the requester. The latter, usually, is interested by the final aggregated answer. Consequently, in our work, we assess the quality of the targeting by computing the accuracy of the aggregation process. That is, the ratio of correct answers among the estimated ones. The correctness can be assessed by comparing the estimated answers with the ground truth answers.

To measure the performance of a selection method, we define the *Accuracy of Crowdsourcing* metric (AoC), to which equivalent quality measures have been used in other works such as in [76, 82, 89]. AoC of an aggregation method *agg* is formally expressed as shown in Equation 5.1 :

$$AoC_{agg}(\mathbb{G}) = \frac{1}{|\mathbb{G}|} \sum_{t \in \mathbb{G}} I(\hat{r}_t^{agg}, r_t) \tag{5.1}$$

Where t is a task in a task set $\mathbb{G}$, r_t is the correct answer of t, $\hat{r}_t^{Agg}$ is the answer estimated by the aggregation method *agg* for the task t, and $I(x,y)$ is the indicator function that we recall here:

$$I(x,y) = \begin{cases} 0 & \text{if} \quad x = y \\ 1 & \text{otherwise} \end{cases} \tag{5.2}$$

5.1.2 Overhead measurement

The Time and Cost of a Crowdsourcing campaign, that we denote *ToC* and *CoC* respectively, are directly related to the number of assignments made during the campaign. One possible estimation of their values can be obtained by multiplying the number of assignment by the time-per-assignment-per-worker *TA*, i.e., the sequential time of completing a task by a worker (explained further) and the reward-per-assignment-per-worker *RA*, i.e., the average price payed to complete one task by one worker. That is :

$$ToC = TA \times Assignment \tag{5.3}$$

$$CoC = RA \times Assignment \tag{5.4}$$

Equations 5.5 and 5.6 compute an estimate of the difference between two different quality control methods i and j. Those reflect the additional assignments made to serve only the quality control process e.g. the probing assignments, test questions, etc. Let us assume that $Assignment_i$ and $Assignment_j$ are the total number of assignments for the method i and j respectively. *RA* is a task dependent parameter, thus, it is similar for both methods when applied on the same task.

TA, on the other hand, is a more complex parameter. In practice, it is very difficult to model how the workers select a certain task, how long they will take to finish it, how many among them will leave without finishing the task and after how much time, how long would it take to get these workers replaced etc. Furthermore, all this happens in a quasi-parallel manner dependent on the configuration of the system at a given moment. That is the workers (number, preferences, qualifications ...) and the tasks (number, rewards, ...) available in the platform at this moment. If all the workers had to start working on the tasks simultaneously, then, the overall time for completing the campaign would be equal to the longest time per

worker. This is the lower bound of ToC. In the opposite, if workers had to perform the tasks in a sequential way where one worker starts completing them when an other finishes doing it, the total time for the task completion would be the sum of all the individual completion times. This is the upper bound of ToC. However, as we detailed earlier, the system is quasi parallel. Thus, in these experiment, we approximate TA by dividing the observed time of task completion by all the worker i.e. the time between submitting the tasks and receiving all the contributions, by the number of assignments achieved during the campaign. Indeed, this approximation is not generic. Its purpose is to allow computing the overheads that would be induced by one quality control method for the configuration of the considered dataset. Finally, we assume that for the same online workers, and for a fixed reward (i.e. one task is as attractive as the other), the time that a task stays running in the system is the same for 2 identical tasks when crowdsourced simultaneously. Hence, $TA_i = TA_j$. Consequently:

$$\begin{aligned} G_{time} = ToC_i - ToC_j &= TA_i \times Assignment_i - TA_j \times Assignment_j \\ &= TA \times (Assignment_i - Assignment_j) \end{aligned} \tag{5.5}$$

$$G_{budget} = CoC_i - CoC_j = RA \times (Assignment_i - Assignment_j) \tag{5.6}$$

5.2 General experimental setup

5.2.1 Dataset

The quality evaluations are performed on CrowdED (Chapter 4). CrowdED has been designed to fulfill a larger specification range than needed for our evaluation process. We describe here the configuration of CrowdED retained for this experiment series. We consider only the MCQ tasks[1]. Their number[2] is 926 and they are distributed over 16 subsets i.e., jobs. We consider 400 workers. Those are chosen so that each of them has, at least, 100 contributions in CrowdED (i.e., they participated in almost 10% of the tasks at least). Only 200 of those workers participated in all the tasks. Workers are characterized by their declarative profiles of 12 features among which we selected the 6 most significant[3] ones and encoded them via dummy coding.

The time and budget evaluations are performed on a dataset called "**Knowledge dataset**" that we collected on Figure Eight before collecting CrowdED. and which we describe hereafter: it consists of 60 knowledge MCQ tasks with known ground truth used to estimate the accuracy afterward. The tasks are distributed evenly over three different knowledge domains: sports, botany and technology. We asked 140 users to answer all the questions. Hence, the resulting dataset consists of 8400 contributions. Average worker accuracy is equal to 61.3%, the MV aggregation accuracy is 80%. The EM aggregation accuracy is equal to 81.3%.

[1] We did not consider free text questions of CrowdED.

[2] Tasks with multiple questions were split into multiple tasks to comply with the definition we used to formalize CAWS.

[3] p-value less than 0.1 in the null hypothesis test.

5.2.2 Ground truth

Computing the accuracy as shown in the previous section supposes that the ground truth answers of all the tasks are already known. For CrowdED, which will be used as the quality evaluation dataset for our work, this ground truth does not exist. In order to estimate the correct answers we relied on the high level of redundancy in CrowdED where each task is labeled at least 280 times. In fact, it has been shown that the majority voting aggregation, yields near perfect estimation of the ground truth (up to 95% of of estimated answers are correct) when the number of workers is very large (greater than 150) [102]. Accordingly, the ground truth answers for CrowdED was estimated through a Majority Vote aggregation of the collected contributions. Those answers are used to estimate the accuracy of the worker and of the aggregation processes when needed.

5.2.3 Selection methods

In this work, two variants of CAWS are compared to a state-of-the-art selection method. As detailed in Chapter 3, the offline learning of CAWS relies on Linear Regression to find the correlations between the workers profiles and their performance for each task type. Indeed, it is possible to use another model to learn these correlations. Besides the Linear Regression based learning, which we denote *CAWS - LR*, we also implement a Random Forest based version of CAWS and we denote it *CAWS - RF*. Both versions are compared, it terms of quality, with the method proposed by Li[4] et al. [74] as it is the state of the art method when it comes to selection method which are agnostic to a priori knowledge and that relies on declarative profiles to target workers. We denote this method *Li - n* where n is the number of tasks used for the probing stage. Two values for n were used: $n = 5$ which is the value used in [74] and $n = 10$ which is a more quality optimistic version of Li's approach. In addition, we use for reference a *Random* selection method where the needed number of workers is randomly picked from the crowd in an uniform manner. This allows to compare the selection methods to a scenario where no selection is made under the same budget constrained (defined by the number of requested contributions ρ_t explained in Section 3.1.2 of Chapter 3). Table 5.1 summarizes these methods and their descriptions.

Method name	Descritpion
CAWS - LR	CAWS with a Linear Regression based discovery algorithm
CAWS - RF	CAWS with a Random Forest based discovery algorithm
Li - n	Li et al. method where n is the number of tasks used for the probing
Random	A random selection process for reference

Table 5.1: A summarized view of the selection methods used in our evaluations.

5.2.4 Task grouping

Task grouping is a key step of CAWS and needs to be evaluated. In our evaluation, we used 2 main grouping methods. First, we used the subsets to which the tasks belonged when the

[4]This method has been proposed by Li and Yu. For simplicity, further in this chapter we denote this method by Li's method.

task corpus has been constituted. A description of these subsets can be found in Table 4.3 of Chapter 4. Using this grouping is equivalent to relying on the requester submission of a job. That is, all tasks submitted in a same job are considered to be similar which is the assumption of Li's method. Last but not least, we used the task clustering approach which is the core of CAWS. Here we used four variants which are the combinations of two clustering algorithms, K-means and agglomerative clustering, and two vectorizing algorithms, TFIDF and Doc2vec. The description of those can be found in Table 5.2.

Method name	Description
ad	Agglomerative clustering and Doc2vec vectorizing
at	Agglomerative clustering and TFIDF vectorizing
kd	K-means clustering and Doc2vec vectorizing
kt	K-means clustering and TFIDF vectorizing
set	CrowdED subsets

Table 5.2: A summarized view of the grouping methods used in our evaluations.

Hereafter, the following annotation is used to reflect the configuration used to test a selection method:

$$\overbrace{\underbrace{A}_{\text{Selection}} - \underbrace{B}_{\text{Selection config.}} - \underbrace{C}_{\text{Grouping}}}^{\text{Method Name}} \quad \text{e.g.} \quad CAWS - LR - kd \quad \text{or} \quad Li - 10 - set$$

For instance, *CAWS - LR - kd* indicates a configuration where CAWS is used for the selection, and that it has been trained over the tasks clustered using K-means applied on Doc2vec task representation and through a Linear Regression. *LI - 10 - set*, means that Li et al. method is used for the selection and that it is applied on CrowdED subsets while using 10 tasks for the probing phase.

5.2.5 Aggregation

From the perspective of the output type[5], tasks in a given cluster are most likely to be heterogeneous for two reasons: first, because the resulting clusters and their intra-homogeneity w.r.t. this aspect, depends on the used features. Second, even if a suitable feature, e.g., the output type, is used to split the tasks, no clustering algorithm is able to yield perfect results. Therefore, one needs to use an aggregation algorithm that is capable of aggregating the tasks independently. EM aggregates a set of tasks at once, thus it is not suitable. Majority voting and weighted majority votes are more convenient as they aggregate the tasks independently (i.e., one task at a time). In the remainder of this chapter, when CAWS is used with its clustering component, the MV algorithm is used to aggregate the contributions of the selected workers and the reported accuracy values are those of MV. Moreover, every time Li's method is used, the reported measures are those of the EM algorithm as their discovery algorithm is optimized for this measure (Section 2.2).

[5]That is, of what type are the contributions? An option among proposed ones, a rating, a free text label, etc.

5.3 Experiments

Now that the general framework of our evaluations is set, we detail in the following paragraphs the experiments that we conducted over two axis: (i) the quality of the learning and selection approach and (ii) the time and cost gain.

5.3.1 Evaluating the performance of CAWS

5.3.1.1 Objectives

Here we aim at comparing the accuracy of CAWS in terms of the aggregation accuracy stated in Equation 5.1. We conducted four experiments Exp.1, Exp.2 , Exp.3 and Exp.4, in which we are interested in comparing the impact of the following parameters on accuracy :

Exp.1 *The Learning Model*: As explained earlier in Section 5.2.3, it is possible to use different learning models in the offline learning phase. In the first experiment, we aim to study the impact of the used learning model on the targeting quality.

Exp.2 *The Aggregation technique*: In our second experiment, we are interested in measuring the impact of the used aggregation algorithm on the targeting process.

Exp.3 *The Probing Tasks*: One benefit of CAWS over other probing based methods is that it leverages the history which provides a larger learning corpus when compared to the limited number of probing tasks. This allows a better representation of the aggregated performance correlated to each profile feature. In this experiment, we intend to measure the impact of the learning corpus size on the targeting quality.

Exp.4 *The Grouping Technique*: Providing a larger corpus of probing tasks is an important advantage of CAWS. However, this is conditioned by CAWS' ability of creating a homogeneous learning corpus. Our solution, explained in Chapter 3 consisted in using a clustering step to group the tasks in the system history. Intuitively, the learning and targeting quality are directly influenced by the quality of this grouping. In the fourth experiment, we study the impact of various grouping techniques on the targeting quality.

5.3.1.2 Experimental setup

To test the performance of CAWS, our experiments were run as follows: for a given set S of tasks e.g. a subset, a cluster, etc., a 4-folds cross validation was conducted. In each fold, S is split into two part: S_l of size $\frac{3\times|S|}{4}$ used for training and S_t of size $\frac{|S|}{4}$ used for testing. The contributions of 300 workers were randomly sampled and used as the system history (i.e., the contribution matrix). This ensures that the model is not overfitted and allows us to consider the case where the offline worker set is not identical to the online worker set.

The training consisted in running the discovery algorithm on S_l. Then, the computed model was tested by running the targeting algorithm for each task in S_t to select a number λ of workers. We then computed $AoC_{agg}(S_t)$ for the selected workers' contributions, where *agg* is the used aggregation algorithm. Moreover, for each fold, Li's method was tested by sampling n tasks from S_l. Those served as probing tasks and the targeting was tested over S_t by selecting the same number λ of workers. This process was repeated 20 times and the average accuracy was reported. A number of worker similar to the one used for

CAWS and Li in the testing was also randomly selected and the accuracy of this selection was also reported for reference. Hereafter, we report the configurations that are specific to each experimental objective explained in the previous subsection:

Exp.1 *The Learning Model*: we computed the average AoC_{EM} and AoC_{MV} for *CAWS - LR - set* and *CAWS - RF - set*.

Exp.2 *The Aggregation technique*: we compare the average AoC_{EM} and AoC_{MV} for *CAWS - LR - set* computed in Exp.1. Here, the *set* option is used to assure this comparison in the condition of an optimal level of group homogeneity.

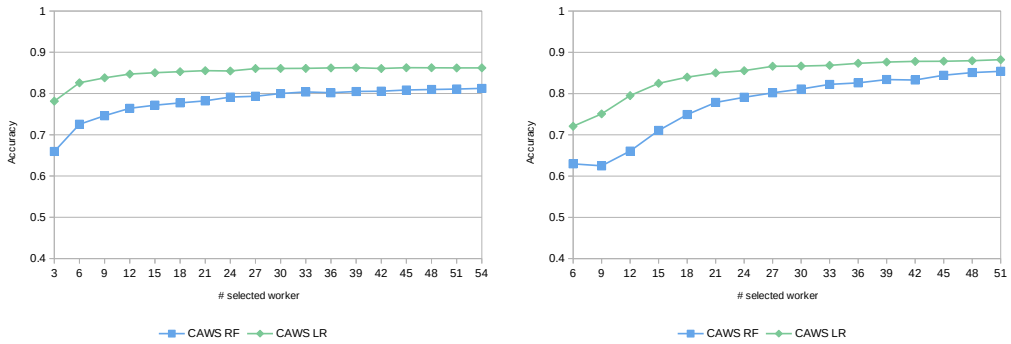
Exp.3 *The Probing Tasks*: we computed the average AoC_{EM} for *CAWS - LR - set*, *Li - 10 - set*, *Li - 5 - set* and *Random - set*. Performing CAWS - LR - set is equivalent to applying Li's method with greater number of tasks. Practically, probing with the whole task set is useless from a selection perspective. And probing with a very large subset of the task is very costly and time consuming.

Exp.4 *The Grouping Technique*: Tasks were clustered into 15 clusters for each of the clustering algorithms. We computed the average AoC_{MV} for *CAWS - LR - set*, *CAWS - LR - ad*, *CAWS - LR - at*, *CAWS - LR - kd*, *CAWS - LR - kt*. *Li - 10 - set* and *Li - 5 - set* are used for comparison.

5.3.1.3 Results

Exp.1 The impact of Learning Model

Figures 5.1a and 5.1b depict, respectively, the accuracy MV and EM w.r.t. the number of selected workers when CAWS is trained with a Linear Regression model and a Random Forest. For both aggregation algorithms, Linear Regression performed better then Random Forest especially for lower selection rates. In the remainder of this chapter, the LR variant of is CAWS is used.



(a) (b)

Figure 5.1: The accuracy of (a) MV and (b) EM w.r.t. the number of selected workers for CAWS - LR and CAWS - RF

Exp.2 The impact of Aggregation technique

Figure 5.2 allows to compare the accuracy of EM and MV when applied on the contributions of the workers selected by CAWS. It shows that, for lower selection rates, MV performs

better than EM. This might be due to the theoretical characteristics of the EM algorithm which makes it less stable for low numbers of workers [102]. When a sufficient number of workers are selected (> 21), EM starts slightly outperforming MV.

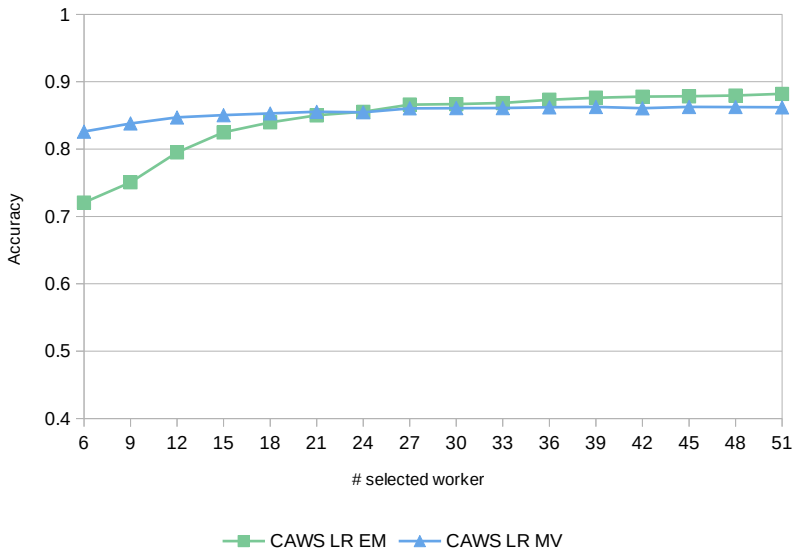


Figure 5.2: A comparison of the accuracy two aggregation algorithms EM and MV w.r.t. the number of the workers selected by CAWS.

Exp.3 The effect of probing set size

Figure 5.3 reports the accuracy of the MV aggregation process for the considered configurations. The subsets of CrowdED are used as task groups and the shown accuracy is the average accuracy of all the subsets. The smallest subset contains 40 tasks, this means that at least, CAWS is trained over 4 times more tasks than Li's method when 10 tasks are used for probing and 8 times when 5 tasks are used for that purpose. This clearly influences the targeting quality as CAWS, in this case equivalent to *Li - >30 - set*, outperforms both *Li - 10 - set* and *Li - 5 - set*. Moreover, it is interesting to mention that, the observed improvement between *CAWS - LR - set* and *Li - 10 - set* is less significant than the difference between *Li - 10 - set* and *Li - 5 - set* despite the fact that the increase in the number of tasks used for probing is greater in the former case than in the latter. To explain this, one needs to observe the accuracy for the individual task sets. That is the accuracy of selecting tasks for only one set at a time i.e. using only the model learned for this set. Figure 5.4 reports the accuracy of the considered methods for a sample of the individual tasks sets. When a task is rather easy (Figures 5.4a and 5.4b), which is reflected by the random selection quickly converging to as high as the accuracy achieved by the non random selection methods, the number of tasks used for the probing is not important. Meanwhile for more difficult tasks (Figures 5.5a and 5.5b), where random selection stays low in quality even for high selection rates, the more tasks are used in the probing stage the better the quality gets. This proves the importance of leveraging the history of the crowdsourcing system as a better solution than online probing, to learn worker performance based on their profile.

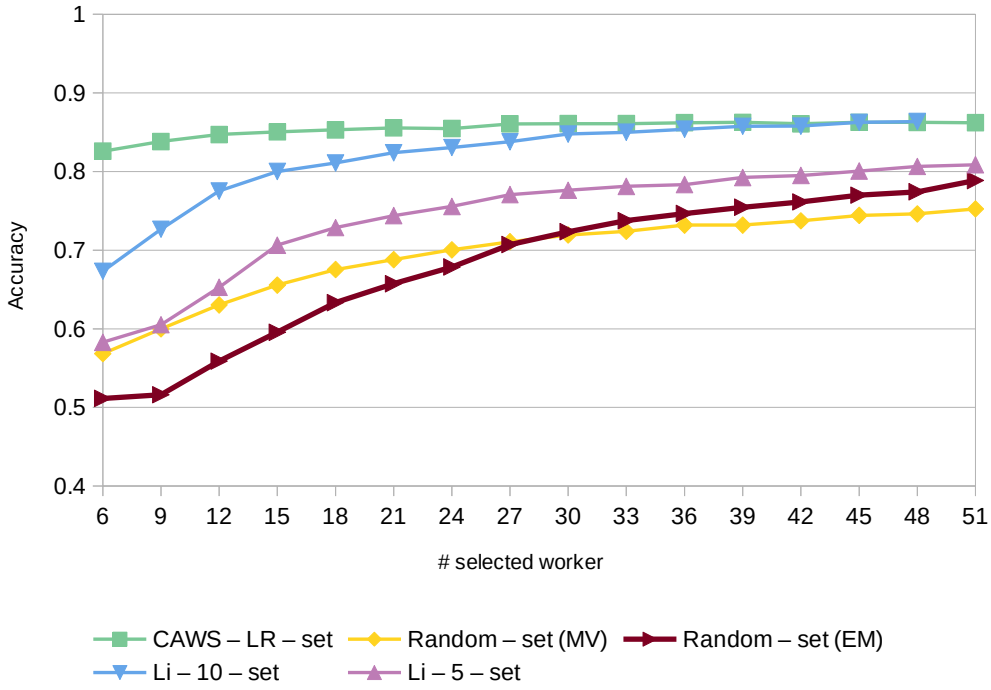


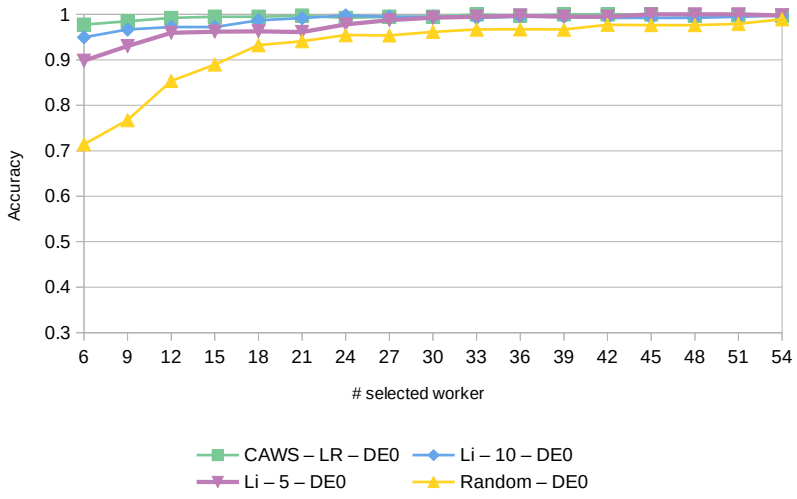
Figure 5.3: Accuracy of the aggregation process using MV w.r.t. the number of workers selected by CAWS, compared to the accuracy for: Li with 5 probing tasks, Li with 10 probing tasks and a random selection process.

Exp.4 The effect of the grouping technique

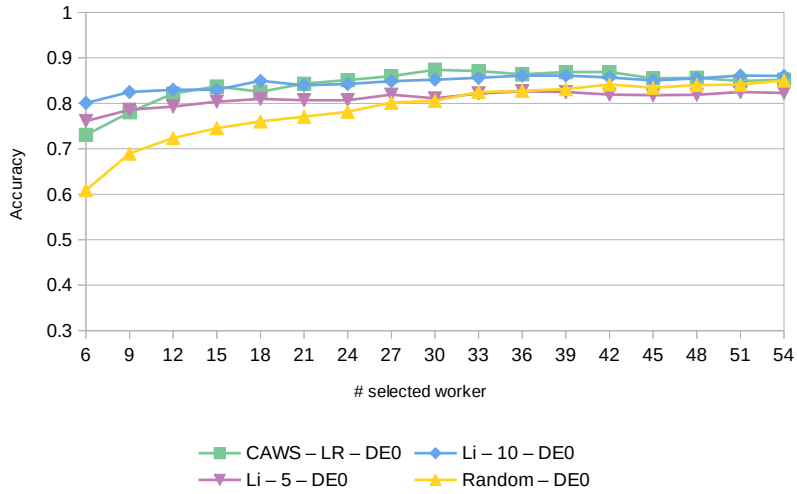
In this paragraph the main results of this evaluation process are discussed. For the previously discussed evaluation results, the grouping component of CAWS was not tested as the used task grouping consisted of the initially available subsets (to allow a better study of the other evaluation parameters). Indeed, using the task sets as groups to learn constitutes the best case scenario of task grouping. That is because tasks proposed in the same job are as homogeneous as it can get. At least, in the common practice in crowdsourcing platforms. Here, we test the possibility of automatically grouping the tasks which makes two things possible : on one hand, this allows to use the similarity with historical tasks to eliminate the online probing phase. On the other hand, it allows to reduce the work load on the requester side who will be able to propose heterogeneous tasks in one job and get them automatically sorted and assigned to suitable workers by the platform.

Figure 5.6 depicts the accuracy of MV in aggregating the contributions of the selected workers when the tasks are grouped through clustering by the offline learning module of CAWS. Here one can make the following observations:

1 One can do a clear separation between two groups of curves in this figure. First, the Doc2vec based vectorizing, and second, the TFIDF based clustering and the set grouping. The former group performs weaker than the latter, that is TFIDF is better in grouping the tasks than Doc2vec. This applies for both clustering algorithms, i.e., K-means and agglomerative clustering. For the same clustering algorithm, TFIDF performed significantly better than Doc2vec (between 7 and 11% over the various



(a) DE0



(b) BI0

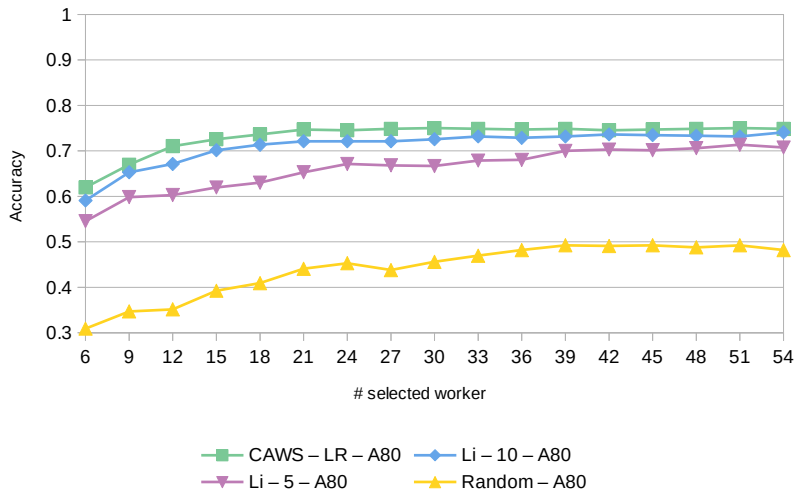
Figure 5.4: The accuracy of the aggregation process applied on the contributions of workers selected by CAWS, Li and random for individual subsets of the dataset: (a) DE0 and (b) BI0.

selection rates).

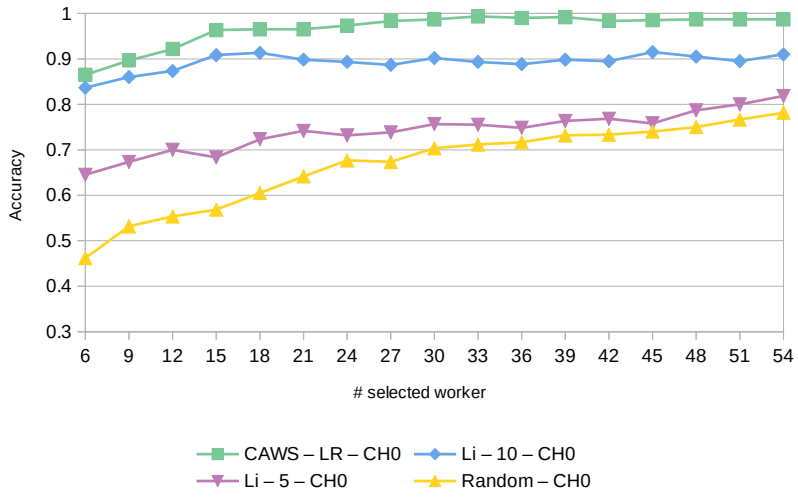
2 For the same vectorizing algorithm, K-means performed marginally better than the agglomerative clustering (between 2 and 4% over the various selection rates).

3 For TFIDF, the accuracy of the agglomerative clustering based CAWS tends, for high selection rates, to the accuracy achieved by learning of the tasks subsets. Meanwhile, the accuracy of K-means clustering based CAWS reached the accuracy of the subset based learning at a lower selection rate (equal to 18). After a selection value of 39 the clustering based algorithm starts improving over the subset based one.

According to observation 1 and 2, it is clear that the features used to vectorize the tasks are a more important parameter than the used clustering algorithm. Two possible reasons



(a) A80



(b) CH0

Figure 5.5: The accuracy of the aggregation process applied on the contributions of workers selected by CAWS, Li and random for individual subsets of the dataset: (a) A80, (b) CH0.

for TFIDF performing better than Doc2vec. First, the task variability in the current iteration of CrowdED might not be sufficient which means that the tasks from different subsets have few common words which means that they will have quite different TFIDF vectors, which in turn, allows them to be easily separated by any clustering algorithm. Second, since the majority of the tasks are short documents, Doc2vec, might not be capable of learning correct models. In which case other variation of the paragraph to vector embedding that are more suited for short documents must be used; Tweet2vec can be a suitable candidate [17].

Now, we compare the performance of the clustering based learning of CAWS with the probing based method of Li. We retain our worst model i.e., CAWS - LR - ad, and our best one, i.e., CAWS - LR - kt, and we compare them with Li's method with 5 and 10 probing tasks. Figure 5.7 shows the result of this comparison. For lower selection rate, the two

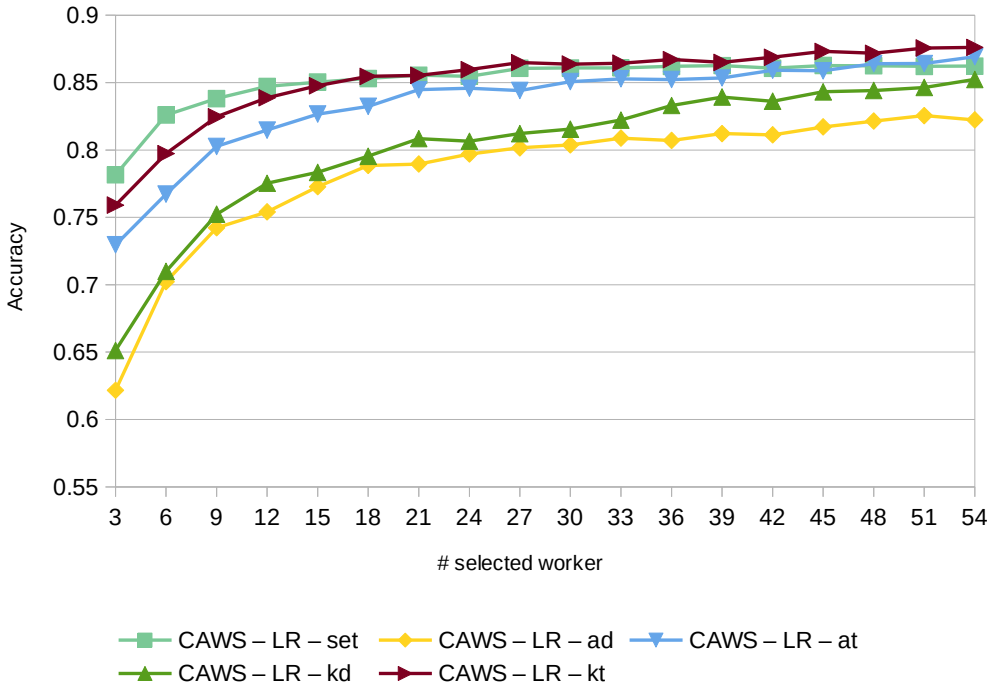


Figure 5.6: A comparison of the accuracy of the MV aggregation processes w.r.t. the number of the workers selected by CAWS for different clustering combinations and a set based grouping.

models of our algorithm outperform Li's method (up to 7% of improvement for CAWS - LR - ad and up to 15% for CAWS - LR - kt). However, for 15 selected workers and more, the 10-probing task based learning of Li starts converging to a level ($Acc = 86\%$) of which our weakest model (Agglomerative clustering and Doc2vec vectorizing) falls short. The model based on K-means clustering and TFIDF, on the other hand, outperforms both variants of Li's method. This is true until the number of selected workers reaches 36, after which the improvement induced by CAWS is marginal.

The quality experiments allowed use to prove that using the tasks in the system history, and grouping them with automatic clustering algorithms, is able to yield a selection quality which in worst case is equivalent to the quality yielded by methods based on online probing. In the next part of this section, we study the ability of CAWS to reduce the time and cost overhead introduced by the quality control process.

5.3.2 Evaluating the time and the budget gain

5.3.2.1 Objective

Now that we showed that our method improves the quality of the selection, we study its ability in reducing the time and budget overheads. For this reason, we conducted two experiments of which the objectives are shown below:

Exp.5 *Overheads estimation*: This experiment aims at computing the time and budget gain induced by CAWS when compared to Li's method.

Exp.6 *Budget study*: This experiment aims at evaluating the output accuracy that we can

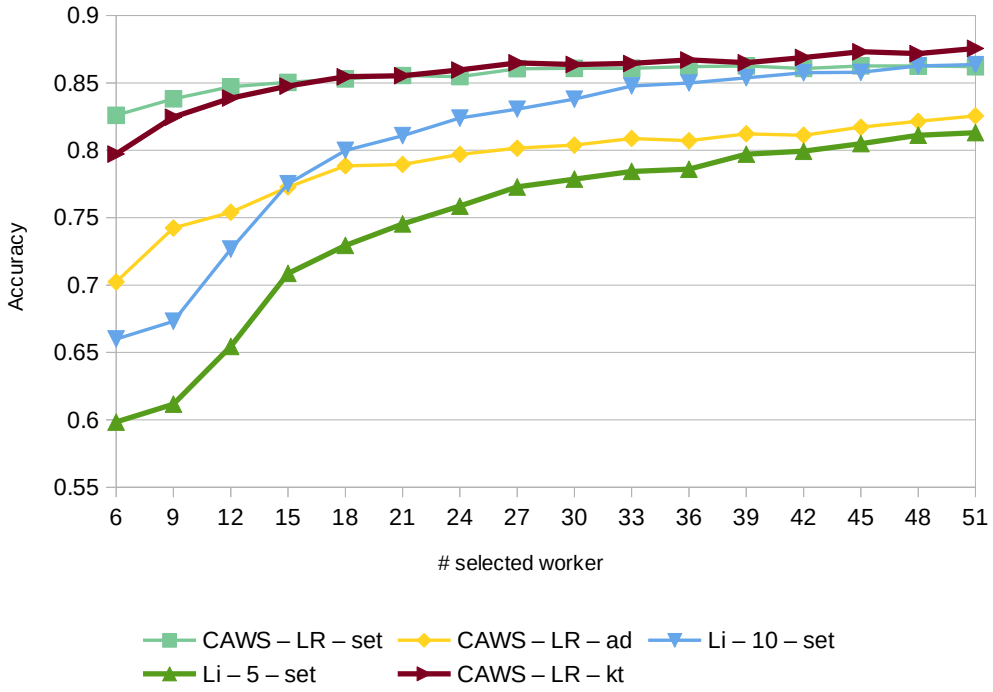


Figure 5.7: A comparison of the aggregation accuracy processes w.r.t. the number of the selected workers by CAWS (clustering) and by Li.

achieve for a fixed budget. Indeed, the concrete known parameter for a random requester is the budget they are planning to spend on their crowdsourcing campaign. Therefore, it is interesting to assess the quality that can be achieved for a given budget.

5.3.2.2 Experimental setup

For the following experiments, we used a smaller dataset - "Knowledge dataset" - as the aim is not to compute quality measures but rather to have an estimation of the overheads for a given selection method.

Exp.5 *Overheads estimation*: We use the observed time and budget spent on completing the campaign which was performed to collect the knowledge dataset, in order to estimate the values of TA and RA. Then, we compute the time and budget gain i.e., G_{time} and G_{budget} of two methods CAWS - LR - all and Li - 10 - all. Here "all" indicates that we are considering the whole dataset as one cluster. Since the objective is to study the accuracy that one can achieve for a given budget, we are not focusing on the clustering aspect.

Exp.6 *Budget study*: For a given number λ of workers to select, we compute the accuracy of MV for the workers selected by CAWS - LR - all and Li - 10 - all w.r.t. the number of tasks used for probing purposes.

Before going into the details of the experiments, we start by computing the number of assignments made during a given campaign. Let $\mathbb{W}_o$ be the set of connected workers and T the set of incoming tasks.

Li : Let γ be the number of tasks used in the learning phase. Let λ_{li} be the number of workers to be selected. The number of assignments χ made during a campaign is given by:

$$\chi_{li} = |\mathbb{W}_o| \times \gamma + \lambda_{li} \times (|T| - \gamma) \tag{5.7}$$

That is, the sum of probing and targeting assignments. Indeed tasks used for the probing are not crowdsourced again as their answers can be inferred from the probing contributions.

CAWS : Let λ_{CAWS} be the number of workers to be selected. The number of assignments χ_{CAWS} made during a campaign is given by:

$$\chi_{CAWS} = \lambda_{CAWS} \times |T| \tag{5.8}$$

For a similar number $\lambda = \lambda_{CAWS} = \lambda_{li}$ of workers to select, the budget gain G_{budget} (Equation 5.6) difference between Li and CAWS is:

$$\begin{aligned} G_{budget} &= RA \times (Assignment_{Li} - Assignment_{CAWS}) \\ &= RA \times (\chi_{li} - \chi_{CAWS}) \\ &= RA \times (|\mathbb{W}_o| \times \gamma + \lambda \times (|T| - \gamma) - \lambda \times |T|) \\ &= RA \times (|\mathbb{W}_o| \times \gamma - \lambda \times \gamma) \\ &= RA \times \gamma \times (|\mathbb{W}_o| - \lambda) \end{aligned} \tag{5.9}$$

Similarly, the time gain G_{time} (Equation 5.5) difference between Li and CAWS is:

$$\begin{aligned} G_{time} &= TA \times (Assignment_{Li} - Assignment_{CAWS}) \\ &= TA \times \gamma \times (|\mathbb{W}_o| - \lambda) \end{aligned} \tag{5.10}$$

Equations 5.9 and 5.10 reflect the cost and the time relative to a number of assignments equal to the number of probing tasks multiplied by the number of workers probed yet not selected. This value is always positive. It equals zero in 2 cases when all the probed workers are selected, or when $\gamma = 0$ in which case the selected workers are randomly selected. In both cases, the point of selection is missed. Moreover, the larger the crowd to probe is i.e., $\mathbb{W}_o$, the larger the time and budget gain get. In fact, this shows that by removing the online probing stage and using offline learning, one can remove the time and budget relative to the quality control process while maintaining the advantages of this step.

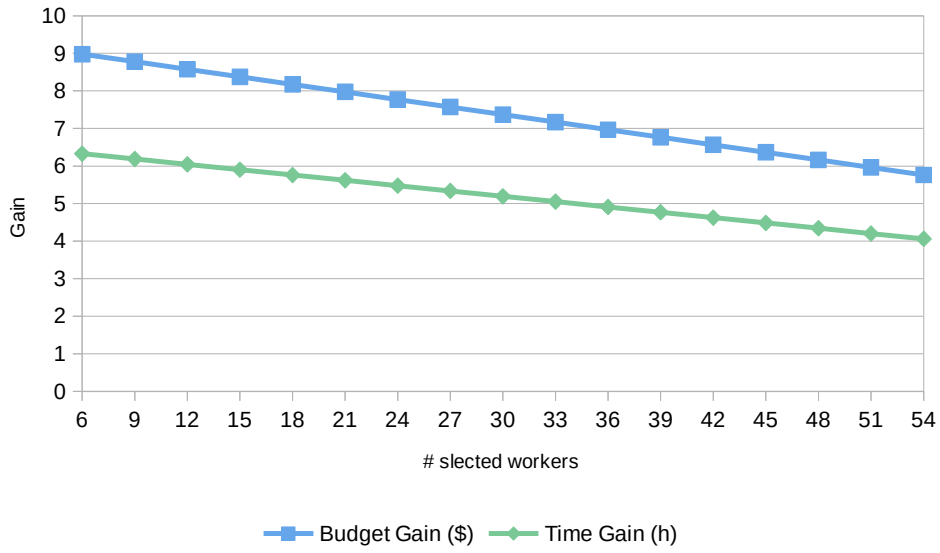


Figure 5.8: Time and budget overheads.

5.3.2.3 Results

Exp.5 Overhead estimation

We start by approximating TA and RA. For the knowledge dataset, we paid 0.4$ per worker[6] for solving all of the 60 tasks. This is equivalent to 0.0067$ per assignment i.e., $RA = 0.0067\$$. The same reasoning is followed to evaluate the time gain; experimental measurements yielded a value of 17 seconds/per assignment i.e., $TA = 17$ sec.

Considering these estimations, we fix the number of probing tasks to 10 and use all 140 workers as the crowd. We compute using equations 5.9 and 5.10, the budget and time gain that CAWS can achieve as compared to the Li's method for $6 < \lambda < 54$. Results are reported in Figure 5.8. The budget and time gain that we achieve is maximal for lower selection rates (e.g. 9$ for $\lambda = 6$) and decreases linearly when the selection rate increases (5.7$ for $\lambda = 54$). This is equivalent to 10 to 16% of the total cost of assigning all 60 task to all 140 workers (i.e., $0.0067 \times 60 \times 140 = 56\$$). That is because budget and time gain relies mainly on removing the probing assignments. When the number of those assignments is important w.r.t. the targeted assignments i.e., for lower values of λ the gain is high. When the number of targeted assignments grows, the relative cost of probing assignments (w.r.t. the whole campaign cost) are reduced. However, since the accuracy of our method is high for lower selection rate, this means we are able to eliminate the cost overheads.

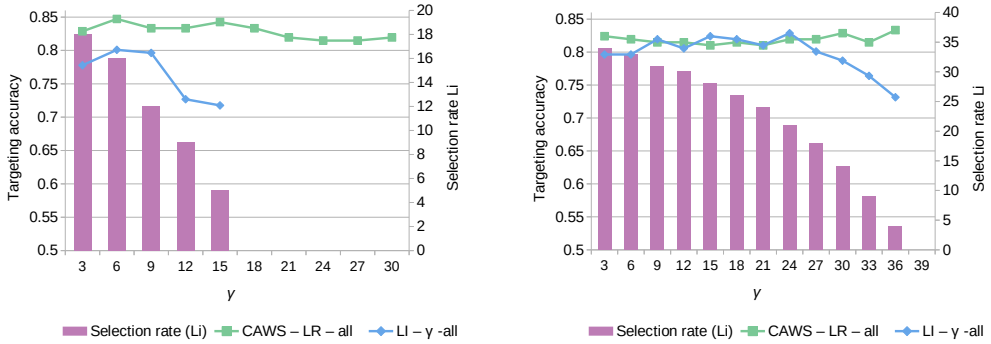
The same applies on the time overhead. Figure 5.8 shows that the time gain drops linearly with the number of workers to select. Yet, at lower rates it is equal to 6 hours. Which means that for the same configuration of the Knowledge dataset, CAWS can allow the collection step to run 6 hours faster than an online probing based method.

[6]Workers who participated to the campaign graded the reward, in average, with 3.8/5 which means that it is fair without being too high w.r.t. the task. Thus it can reflect a good estimation of an average acceptable reward in a crowdsourcing platform.

Exp.6 Budget study

This experiment aims at evaluating the output accuracy that we can achieve for a fixed budget. We compared this accuracy with the accuracy achieved by Li for the similar budget. In fact, the budget in our method is directly reflected by the selection rate (λ). Whilst, the budget in Li method depends not only on its selection rate (that we denote λ_{li}) but also on the size of the crowd to probe as well as the split of the task used for probing, Li uses a subset of the tasks to probe the crowd. This probing can indeed be done on the whole crowd or a part of it. For a similar budget, the maximum number of assignments that can be done by both methods is identical. Hence :

$$\chi_{Li} = \chi_{CAWS} \Leftrightarrow |\mathbb{W}_o| \times \gamma + \lambda_{li} \times (|T| - \gamma) = \lambda_{CAWS} \times |T| \tag{5.11}$$



(a) (b)

Figure 5.9: EM accuracy for a fixed budget: (a) $\lambda = 21$ and (b) $\lambda = 35$ (Exp.6)

We consider that λ_{CAWS} and λ_{li} are not necessarily equal. Figure 5.9a shows the results for $7 < \gamma < 30$ and a fixed budget determined by $\lambda = 21$. When a larger split of the tasks is used in the probing stage, a smaller number of workers can be targeted. For $\gamma = 18$ the whole budget is spent on probing the crowd. Hence, it is not possible to select workers. For $\gamma < 18$, the quality of the learning grows until the number of targeted workers becomes very low (for $\gamma = 12$, $\lambda_{li} = 8$) and thus, the aggregation quality starts dropping to 71%. Our method has an average accuracy of 82%. Figure 5.9b shows the results for $\lambda = 35$. That is 35 workers can be selected by our method. For this budget our method is as good as Li's method for $9 \leq \gamma \leq 24$, it has an average accuracy of 81%. Li et al.'s accuracy is low for $\gamma < 6$ because 3 tasks are not enough to probe the crowd and for $\gamma > 33$ because of the low selection rates. Being independent from any probing stage, our method uses the whole budget to target workers. This budget can be equal to the minimum value for which the collected contributions are enough to allow a stable convergence of the aggregation method [100].

5.4 Summary

In this chapter we evaluated CAWS and compared it to an online probing based selection method in terms of quality and time and budget gain. Our findings can be summarized as

follows:

The quality evaluation showed that it is possible to substitute online probing stages by an offline learning step to learn about the correlations between workers profiles and their performances for different task types. Indeed, this is enabled by the ability of clustering, as shown in the results, to deliver an adequate task grouping which translates in good learning and selection accuracy. Moreover, the experiment showed that offline learning had an important advantage over the online probing as it allowed a better estimation of the worker accuracies and thus a better learning and targeting processes. This jump in the accuracy estimation quality is guaranteed by a learning corpus larger than the few probing tasks.

Overhead study showed that CAWS is able to reduce the time and the budget of completing a crowdsourcing campaign. Indeed, while in the experiment we used Li's probing as an adversary method to show the advantage of the offline learning, we could have used any other test question and quiz test based methods as they introduce a similar overhead as the one introduced with Li. Reducing the overhead can be translated in that one can achieve a better (or equivalent) quality than existing quality control method, with a smaller budget and a shorter time.

6

Conclusion and perspectives

6.1 Conclusion

To conclude this work, we start by summarizing the workflow we followed in presenting it. In the first chapter of this thesis, we introduced the quality issue in crowdsourcing and argued on the necessity of quality control mechanisms. In Chapter 2, we reviewed the literature of these mechanisms and discussed it in terms of three criteria: the time overhead, the cost overhead and the achieved quality. This discussion backed up the adoption, in this thesis, of a worker selection approach. Then, in Chapter 3, we introduced CAWS, which is a worker selection method that aims at controlling the quality of the crowdsourcing output, while reducing the time and cost overhead resulting from this control. In Chapter 4, we presented CrowdED, which is an information rich dataset that contains qualitatively and quantitatively all the needed elements to allow a sound evaluation of CAWS. In addition, we introduce in this chapter an extension and enrichment platform for CAWS called CREX. Finally in Chapter 5, we showed through an experimental evaluation the ability of CAWS to achieve efficient quality control. The contributions of this thesis are summarized in the following paragraphs.

CAWS

Controlling the quality of crowdsourcing campaigns is a mandatory step towards collecting usable and valid data. Many methods have been proposed to achieve this quality control. These suffered from two major limitations: their dependency on a priori knowledge about the individual workers and the additional time and budget overheads they introduce. To overcome these limitation, we proposed a Context Aware Worker Selection method (CAWS). CAWS leverages the crowdsourcing system history to build knowledge about the task types and their relationship with the workers profiles. It operates as follows: to begin, in an **offline** phase, completed tasks are clustered into homogeneous groups for each of which the correlation with the workers declarative profile is learned. Then, in the **online** phase (i.e. the actual crowdsourcing), an incoming task is matched to one of the existing clusters and the

correspondent profile model is used to select the most reliable workers for the given task. CAWS eliminates the need for a priori knowledge about individual workers by leveraging, on the one hand, the declarative profiles of the workers and, on the other hand, the similarity between the tasks. Moreover, it reduces the time and budget overheads by eliminating the need for an online probing step which it replaces by the offline learning phase.

Evaluating CAWS proved that using text vectorizing techniques allows to group the tasks into homogeneous clusters over which the learning process can be applied. Results showed that one can reduce the time and budget overheads by eliminating the probing stage while achieving aggregation accuracies similar to these achieved through a job-based grouping. Term frequency based vectorizing outperformed the more semantic Doc2Vec vectorization which might be due to the short length of the tasks. Besides, this offline learning showed to be beneficial, especially for harder tasks which are usually the most time and budget consuming tasks in crowdsourcing since they require more redundant contributions.

CrowdED

The crowdsourcing research community lacks for information-rich datasets such as the one needed to evaluate CAWS. Indeed, CAWS leverages a multitude of task and worker features in order to complete its selection task. In the available datasets, these features are either absent or insufficiently present. In order to overcome this dataset challenge and evaluate CAWS, we designed and built CrowdED (Crowdsourcing Evaluation Dataset). CrowdED is a rich dataset to evaluate quality control methods and quality-driven tasks vectorization and clustering. The generation of CrowdED relies on a constrained sampling approach that allows to produce a task corpus which respects both, the budget and type constraints. Beside helping in evaluating CAWS, and through its generality and richness, CrowdED helps plugging the benchmarking gap mentioned earlier.

CREX

To maintain its usability as a generic evaluation dataset for quality control methods, CrowdED is meant to be extended and enriched collaboratively. This allows it to fulfill the requirements of future quality control mechanisms. To enable this collaborative extension, we presented CREX (CReate Enrich eXtend), an open source platform for creating, enriching, and extending crowdsourced datasets like CrowdED. Thanks to its configuration panel CREX is an **easy to use** platform. Its **modular** architecture and open source availability allow it to be collaboratively **extended**. Furthermore, these modules, which are highly **configurable** can be used together in an end-to-end fashion, independently or as an entry point of any sub-workflow of CREX. All of this renders the **multipurpose** use of CREX possible.

6.2 Perspectives

In this section we discuss the perspectives of the work achieved in this thesis. We start by a short term perspective which consists in testing an ongoing work on a new feature taxonomy for task characterization. Moreover we describe a feature weighting method that infers the importance of each feature from the historical tasks. Then, we discuss some long term perspectives concerning the worker preferences and task trends handling.

6.2.1 Feature Taxonomy

In its current version, CAWS relies on text document vectorization techniques in order to discover the task contexts. This approach considers mainly the content of tasks. However, crowdsourcing tasks provide more dimensions than the bare knowledge related one. In the literature, many works [1, 3, 80, 96, 107] proposed to characterize tasks through human skills as this facilitates the matching process with the workers characterized in the same feature space. The limitation of this approach raises from the fact that this skill characterization step is done manually by the requesters which is prone to biases and inconsistency and which adds a heavy work load on the requester side. Moreover, rewards, input and output types, length, and other features can also be used to better estimate the similarity between the tasks and thus, allows a better discovery of the task types.

We advocate an automatic feature extraction approach which allows a fine grained and a more dynamic task categorization through low level task features. These features need to be selected and grouped in a way to reflect the human skills and abilities. To this end, we propose to use the well known human abilities taxonomy of Fleishman et al. [34] and to generalize the taxonomy proposed by [43] to get the following five categories of human skills : perceptual, technical, decisional, behavioral and knowledge-related. We then propose to derive three categories of low level task features as follows:

Presentation features describe the layout, language, length of the task, the type of input data (texts, multimedia) and expected answer types (textual, graphical, boolean).

Action features describe the action that must be taken by the worker (give a label, edit a text, tag an image, etc.).

Content features describe the actual content and knowledge domain of a task.

These three types of features reflect for a worker, respectively, her perceptual and behavioral skills [108], her technical and decisional skills and her knowledge in the task topic. Extracting these features is a process of document parsing and text mining for which different techniques can be used as shown in [31]. Indeed, these features are not equally important in describing the task. One can argue, for example, that knowledge and technical skills have a greater influence on the worker performance and thus, features related to these skills must have a greater importance in the clustering process. Therefore features should be be weighted. Here, we seek inspiration in recommender systems and describe a method that finds these hidden weights. We propose, first, to use a linear feature similarity combination. We denote by Sim_A, Sim_C and Sim_P three different similarity measures of action, content and presentation features respectively. A linear combination of these measures is a similarity measure that can be expressed as shown in equation (6.1).

$$\begin{aligned} Sim(t_i,t_j) = \quad & \omega_A Sim_A(t_i,t_j) \\ + \quad & \omega_C Sim_C(t_i,t_j) \\ + \quad & \omega_P Sim_P(t_i,t_j) \end{aligned} \tag{6.1}$$

In equation (6.1), ω_A, ω_C and ω_P are the weights of the different feature types to be found. In order to compute the optimal values of these weights, we can use an item profiling

technique that is common in the collaborative-filtering recommendation. For each task, we compute the individual accuracy of all workers who participated in it. These accuracies are used as features to form a profile vector for the task. After computing all the task profiles, we can build a matrix Π whose columns are task profiles and rows are the accuracy of workers in all tasks. Π is shown in equation (6.3). In fact, the similarity between task in the accuracy space i.e. in Π, is considered as a ground truth since it is derived from real world data. It is then a logical and reliable base for the weight inference process.

Let t be a task.
Let W be a set of workers of size n.
Let α_{wt} denote the accuracy of worker w in answering a tasks t.
π_t denotes the derived profile of t such that:

$$\pi_t = < \alpha_{1t}, \alpha_{1t}, \ldots, \alpha_{nt} > \tag{6.2}$$

Let T be a set of tasks of size m.
Let W be a set of workers of size n.
Π denotes the derived profile matrix of the system:

$$\Pi = \begin{array}{c} \\ w_1 \\ w_2 \\ \vdots \\ w_n \end{array} \begin{array}{c} \begin{array}{cccc} t_1 & t_2 & \ldots & t_m \end{array} \\ \begin{pmatrix} \alpha_{11} & \alpha_{12} & \ldots & \alpha_{1m} \\ \alpha_{21} & \alpha_{22} & \ldots & \alpha_{2m} \\ \vdots & \vdots & & \vdots \\ \alpha_{n1} & \alpha_{n2} & \ldots & \alpha_{nm} \end{pmatrix} \end{array} \tag{6.3}$$

We consider a vector similarity function (e.g. cosine similarity) denoted Sim' and we compute the similarity in Π of all task couples $(t,t') \in T \times T$.

$$f_{th}(t,t') = \begin{cases} 1 & if \quad Sim'(t_i,t_j) > th \\ 0 & if \quad Sim'(t_i,t_j) < th \end{cases} \tag{6.4}$$

Where th a given similarity matrix. Using the found similarities and the decision function shown in (6.4), we can build a linear model that can be solved to estimate ω_A, ω_C and ω_P.

$$f_{th}(t,t') \sim \omega_A Sim_A(t_i,t_j) + \omega_C Sim_C(t_i,t_j) + \omega_P Sim_P(t_i,t_j) + \varepsilon \tag{6.5}$$

6.2.2 Adding worker preference

Recently, few works started exploring the field of worker-centric quality control and throughput optimization methods [1, 96]. Our method can be classified under the category of requester-centric quality control. That is, it takes into account the budget and quality requirements of the requesters in selecting the workers. Yet, it does not consider, explicitly[1],

[1] An implicit assumption about the worker preference is indeed made in our approach as workers are usually good in performing tasks they like.

the preferences and the requirements of the workers who constitute the backbone of the crowdsourcing system. One possible improvement of CAWS towards including the worker preferences in the selection process, is to use a compound quality measure during the learning process instead of a pure quality oriented one. For instance, one can compute a weighted linear combination between the worker accuracies for the historical tasks and his preference score for these tasks. Preference scores can be extracted from implicit e.g., number of tasks solved in the same cluster, average lifespan per task set, etc. or explicit feedbacks e.g., task ratings, requester ratings, etc. This compound accuracy/preference score can be used as targets for the profile observations in the discovery algorithm (Section 3.3.2) to find the correlation between the worker features and their reliability and preference score for a given type of tasks.

6.2.3 Concept shift and model adjustment

The interest in using crowdsourcing keeps growing to reach a wider range of tasks. New task types start emerging due to the need for new input and output types, of action types, of content and so on. This can have an impact on the performance of CAWS. That is because, after a certain lifespan, the trained model might no more suit these new types. It might be possible that incoming tasks can't be matched to any cluster among the precomputed ones. Hence none of the learned models is suitable to target workers for these tasks. It is crucial then to have an update strategy to adapt the computed models. This change in the task type trends is less likely to affect the whole system at once and is more likely to affect specific clusters. Retraining the whole system, besides its computational cost, might be unnecessary. Studying the structure of each cluster and detecting the formation of new type patterns can allow to avoid recomputing all of the models and restrict it to the changing ones only.

6.2.4 Quality guarantees

When evaluating CAWS, we observed a correlation between the agreement of the worker in a given task cluster - measured with the Krippendorff's alpha[2] [66] - and the quality of the selection process. This correlation can be leveraged to propose time and quality guarantees to the requester. That is, for task types where the agreement in the history is very high, the crowdsourcing platform can propose a looser selection process to the requester. This allow a faster contribution collection step. For the other task types where the historical agreement is less obvious, a more strict selection need to be performed and only top workers should be targeted. However, if these are not available at a given moment, the platform can compute a quality guarantee score of the online workers, using the agreement of workers with similar profiles found in the history. Based on this guarantee score, a suitable reward per assignment can be recommended to the requester. The requester can then make the choice between faster and cheaper, yet lower quality contribution collection step and a slower and more expensive, yet high quality contributions.

[2]This is a global agreement metric that measure the how much a set of workers agrees on the answers of a set of tasks by comparing the answer distribution to a random answer distribution.

A

Details about the CREX platform

A.1 Dependencies

CREX code is developped in [Python](https://www.python.org/) v3.5.2. The following packages are used:

1. **scikit-learn** : `http://scikit-learn.org/stable/` (clustering and evaluation measures)
2. **pandas** : `http://pandas.pydata.org/` (data structure and matrix handling)
3. **scipy** : `https://www.scipy.org/` (scientific computing library)
4. **numpy** : `http://www.numpy.org/` (scientific computing library)
5. **seaborn** : `https://seaborn.pydata.org/` (data visualization)
6. **nltk** : `https://www.nltk.org/` (natural language processing)
7. **gensim** : `https://radimrehurek.com/gensim/` (doc2vec implementation)
8. **termcolor** : `https://pypi.python.org/pypi/termcolor` (visual console output)

A.2 References

CREX and CrowdED can be found on Figshare associated to the following DOI:

URL https://figshare.com/s/ca41a59f73c092385fc3

DOI https://doi.org/10.6084/m9.figshare.6109559 (Inactive until resource is made public)

A.3 Configurations

Following is a list of the configurable parameters of CREX along with their descriptions:

Parameter name	Values	Description
preprocess_	[True/False]	Preprocess the data or not, i.e., tokenize/stemm/train the vectorizer models. If False a pre-processed data folder should be given.
vectorize_	[True/False]	Compute the feature vectors or not, e.g., TFIDF transform. If False a pre-processed data folder should be given.
cluster_	[True/False]	Run CM or not.
sample_	[True/False]	Run SM or not.
evaluate_	[True/False]	Run EM or not.
distance_metric_	[euclidean/cosine]	Distance metric to be used by CM.
doc_sample_sizes	[array of size 1]**	e.g. [1000] If a large corpus is used to train the vectorizers, a subsample of this corpus can be vectorized and clustered if needed
preprocess_tfidf	[True/False]	Whether to train the TFIDF vectorizer or not
preprocess_doc2vec	[True/False]	Whether to train the Doc2Vec vectorizer or not
different_custering_data	[True/False]	Whether to use a corpus different from the one used for training the models or not
n_clustering_processes	[integer]	Number of processes for the parallel execution of the clustering.
n_evaluation_processes	[integer]	Number of processes for the parallel execution of the evaluation
kmeans_k_	[array of values/PARI]*	Number of cluster for Kmeans.
minibatch_km_	[0/INTEGER]	If 0 minibatch is not used, else minibatch is run with the given batch size
dbscan_min_points_	[array of values/PARI]*	The minimum point parameter of DBSCAN
dbscan_eps_	[array of values/PARI]*	The EPSILON parameter of DBSCAN
agg_k_	[array of values/PARI]*	Number of cluster for the agglomerative clustering.
agg_linkage_	['ward', 'complete', 'average']	The linkage parameter of the agglomerative clustering.
doc2vec_sizes_	[array of size 1]**	Size of produced Doc2vec vectors
doc2vec_windows_	[array of size 1]**	Size of used Doc2vec window
tfidf_vector_sizes_	[array of size 1]**	Size of produced TFIDF vectors
tfidf_pca_	[True/False]	Whether to use PCA dimension reduction or not
tfidf_vector_sizes_pca_	[array of size 1]	Size of the PCA vector
sampling_fitness	[rmse/minmax]	The objective function of the dampling algorithm
max_sample_size_	[INTEGER]	Size of output sample ("S" in the draft)
min_samples_per_cluster_	[INTEGER]	Minimum sample size per cluster ("th" in the draft)
max_sampling_iteration_	[INTEGER]	Maximum number of itterations ("itt" in the draft)
eva_measures	[array of ('hcv'/'sil'/'coc')]	The evaluation measres to compute by the EM
eva_vectorizing_models	['tfidf'/'doc2vec']	The vectorizing modules to evaluate
eva_clustering_models	['kmeans'/'agg'/'dbscan']	The clustering modules to evaluate
result_folder	[PATH string]	A path to the output folder
raw_data_folder	[PATH string]	A path to the input data folder
preprocessed_location	[PATH string]	A path to the preprocessed data folder
raw_clustering_data_folder	[PATH string]	A path to the data to cluster if (different_custering_data is True)
clustering_vectorizing_combs	[array of ('clustModel_vectModel')]	e.g. ['kmeans_tfidf', 'dbscan_doc2vec'] tells the VM and CM what models to train

References

[1] Maha Alsayasneh et al. “Personalized and Diverse Task Composition in Crowdsourcing”. In: *IEEE Transactions on Knowledge and Data Engineering*. Volume 30. 1. IEEE, 2018, pages 128–141.

[2] Vamshi Ambati, Stephan Vogel, and Jaime G Carbonell. “Towards Task Recommendation in Micro-Task Markets”. In: *AAAI Conference on Human Computation*. 2011.

[3] Sihem Amer-Yahia et al. “Task composition in crowdsourcing”. In: *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE. 2016, pages 194–203.

[4] Apisense. URL: `http://www.apisense.io/`.

[5] Leif Azzopardi, Mark Girolami, and Keith van Risjbergen. “Investigating the relationship between language model perplexity and IR precision-recall measures”. In: *International ACM SIGIR conference on research and development in information retrieval*. ACM. 2003, pages 369–370.

[6] Yukino Baba and Hisashi Kashima. “Statistical quality estimation for general crowdsourcing tasks”. In: *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2013, pages 554–562.

[7] Marko Balabanović and Yoav Shoham. “Fab: content-based, collaborative recommendation”. In: *Communications of the ACM*. Volume 40. 3. 1997, pages 66–72.

[8] Piyush Bansal, Carsten Eickhoff, and Thomas Hofmann. “Active content-based crowdsourcing task selection”. In: *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM. 2016, pages 529–538.

[9] JA Bargh and TL Chartrand. *Studying the mind in the middle: A practical guide to priming and automaticity research. Handbook of research methods in social psychology*. 2000.

[10] Michael S Bernstein. “Crowd-powered systems”. In: *KI-Künstliche Intelligenz*. Volume 27. 1. Springer, 2013, pages 69–73.

[11] David M Blei, Andrew Y Ng, and Michael I Jordan. “Latent dirichlet allocation”. In: *Journal of Machine Learning Research*. Volume 3. Jan. 2003, pages 993–1022.

[12] Steve Cayzer and Uwe Aickelin. *Recommender system based on the immune network*. 2002.

[13] Xi Chen, Qihang Lin, and Dengyong Zhou. “Optimistic knowledge gradient policy for optimal budget allocation in crowdsourcing”. In: *International Conference on Machine Learning*. 2013, pages 64–72.

[14] A. P. Dawid and A. M. Skene. “Maximum Likelihood Estimation of Observer Error-Rates Using the EM Algorithm”. English. In: *Journal of the Royal Statistical Society. Series C (Applied Statistics)*. Volume 28. 1. 1979, pp. 20–28.

[15] Arthur P Dempster, Nan M Laird, and Donald B Rubin. “Maximum likelihood from incomplete data via the EM algorithm”. In: *Journal of the royal statistical society. Series B (methodological)*. JSTOR, 1977, pages 1–38.

[16] Jia Deng et al. “Imagenet: A large-scale hierarchical image database”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2009, pages 248–255.

[17] Bhuwan Dhingra et al. “Tweet2vec: Character-based distributed representations for social media”. In: *arXiv preprint arXiv:1605.03481*. 2016.

[18] Djellel Eddine Difallah, Gianluca Demartini, and Philippe Cudré-Mauroux. “Pick-a-crowd: tell me what you like, and i’ll tell you what to do”. In: *Proceedings of the 22nd international conference on World Wide Web*. ACM. 2013, pages 367–374.

[19] Djellel Eddine Difallah et al. “Scaling-up the crowd: Micro-task pricing schemes for worker retention and latency improvement”. In: *Second AAAI Conference on Human Computation and Crowdsourcing*. 2014.

[20] Djellel Eddine Difallah et al. “The dynamics of micro-task crowdsourcing: The case of amazon mturk”. In: *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee. 2015, pages 238–247.

[21] Anhai Doan, Raghu Ramakrishnan, and Alon Y Halevy. “Crowdsourcing systems on the world-wide web”. In: *Communications of the ACM*. Volume 54. 4. ACM, 2011, pages 86–96.

[22] Amazon Mechanical Turk documentation. *Creating and Managing Qualifications*. accessed 05-03-2018. URL: `https://docs.aws.amazon.com/AWSMechTurk/latest/AWSMechanicalTurkRequester/Concepts_QualificationsArticle.html`.

[23] Figure-Eight documentation. *Guide To: Contributors - Geography/Language Page*. accessed 05-03-2018. URL: `https://success.figure-eight.com/hc/en-us/articles/201855689-Guide-To-Contributors-Geography-Language-Page`.

[24] Figure-Eight documentation. *Guide to: Contributors Page*. accessed 05-03-2018. URL: `https://success.figure-eight.com/hc/en-us/articles/201855679-Guide-to-Contributors-Page`.

[25] Figure-Eight documentation. *Guide To: Running Surveys*. accessed 05-03-2018. URL: `https://success.figure-eight.com/hc/en-us/articles/201855969-Guide-To-Running-Surveys`.

[26] Figure-Eight documentation. *Test Question Best Practices*. accessed 05-03-2018. URL: https://success.figure-eight.com/hc/en-us/articles/213078963-Test-Question-Best-Practices.

[27] Figure-Eight documentation. *Test Question Enabling and Quiz Mode*. accessed 05-03-2018. URL: https://success.figure-eight.com/hc/en-us/articles/211363686-Test-Question-Enabling-and-Quiz-Mode.

[28] Steven Dow et al. "Shepherding the crowd yields better work". In: *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*. ACM. 2012, pages 1013–1022.

[29] Susan T Dumais. "Latent semantic analysis". In: *Annual review of information science and technology*. Volume 38. 1. Wiley Online Library, 2004, pages 188–230.

[30] Facebook. URL: http://www.facebook.com/.

[31] Ronen Feldman and James Sanger. *The text mining handbook: advanced approaches in analyzing unstructured data*. 2007.

[32] FigureEight. URL: http://www.crowdflower.com/.

[33] Ailbhe Finnerty et al. "Keep it simple: Reward and task design in crowdsourcing". In: *Proceedings of the Biannual Conference of the Italian Chapter of SIGCHI*. ACM. 2013, page 14.

[34] E. A. Fleishman. "Toward a taxonomy of human performance". In: *American Psychologist*. 1957, pages 1127–1149.

[35] Freelancer. URL: http://www.freelancer.com/.

[36] Inc. Gartner. *Emerging Technology Hype Cycle for 2017*. accessed 20-03-2018. July 2017. URL: https://blogs.gartner.com/smarterwithgartner/files/2017/08/Emerging-Technology-Hype-Cycle-for-2017_Infographic_R6A.jpg.

[37] David Geiger and Martin Schader. "Personalized task recommendation in crowdsourcing information systems — Current state of the art". In: *Decision Support Systems*. Volume 65. 2014, pages 3 –16.

[38] *Gensim*. https://radimrehurek.com/gensim/. Accessed: 2018-02-25.

[39] Arpita Ghosh, Satyen Kale, and Preston McAfee. "Who moderates the moderators?: crowdsourcing abuse detection in user-generated content". In: *Proceedings of the 12th ACM conference on Electronic commerce*. ACM. 2011, pages 167–176.

[40] Yolanda Gil et al. "A Controlled Crowdsourcing Approach for Practical Ontology Extensions and Metadata Annotations". In: *International Semantic Web Conference*. Springer. 2017, pages 231–246.

[41] Daniel Haas et al. "Argonaut: Macrotask Crowdsourcing for Complex Data Processing". In: *PVLDB*. Volume 8. 2015, pages 1642–1653.

[42] Christopher Harris. “You’re hired! an examination of crowdsourcing incentive models in human resource tasks”. In: *Proceedings of the Workshop on Crowdsourcing for Search and Data Mining (CSDM) at the Fourth ACM International Conference on Web Search and Data Mining (WSDM)*. Hong Kong, China. 2011, pages 15–18.

[43] U. Hassan and E. Curry. “A capability requirements approach for predicting worker performance in crowdsourcing”. In: *IEEE CollaboratCom*. 2013, pages 429–437. DOI: 10.4108/icst.collaboratecom.2013.254181.

[44] Chien-Ju Ho, Shahin Jabbari, and Jennifer Wortman Vaughan. “Adaptive task assignment for crowdsourced classification”. In: *International Conference on Machine Learning*. 2013, pages 534–542.

[45] Thomas Hofmann. “Probabilistic latent semantic analysis”. In: *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc. 1999, pages 289–296.

[46] Jeff Howe. “Crowdsourcing: A Definition”. In: *Wired Blog Network: Crowdsourcing*. 2006.

[47] Jeff Howe. *Crowdsourcing: How the power of the crowd is driving the future of business*. Random House, 2008.

[48] Nguyen Quoc Viet Hung et al. “An evaluation of aggregation techniques in crowdsourcing”. In: *International Conference on Web Information Systems Engineering*. Springer. 2013, pages 1–15.

[49] IBISworld. *Crowdsourcing service providers*. accessed 20-03-2018. 2018. URL: https://www.ibisworld.com/industry-trends/specialized-market-research-reports/advisory-financial-services/outsourced-office-functions/crowdsourcing-service-providers.html.

[50] Indiegogo. URL: http://www.indiegogo.com/.

[51] Innocentive. URL: http://www.innocentive.com/.

[52] Panagiotis G Ipeirotis. *Demographics of mechanical turk*. 2010.

[53] Yuan Jin et al. “Leveraging Side Information to Improve Label Quality Control in Crowd-sourcing”. In: *Human Computation*. 2017.

[54] NL Johnson, SI Kotz, and N Balakrishnan. “Beta distributions”. In: *Continuous univariate distributions. 2nd ed. New York, NY: John Wiley and Sons*. 1994, pages 221–235.

[55] Audun Jøsang, Roslan Ismail, and Colin Boyd. “A survey of trust and reputation systems for online service provision”. In: *Decision support systems*. Volume 43. 2. Elsevier, 2007, pages 618–644.

[56] Hyun Joon Jung and Matthew Lease. “Improving Consensus Accuracy via Z-Score and Weighted Voting.” In: *Human Computation*. 2011.

[57] Evangelos Kanoulas et al. *Overview of the trec 2011 session track*. 2011.

[58] David R. Karger, Sewoong Oh, and Devavrat Shah. “Iterative Learning for Reliable Crowdsourcing Systems”. In: *Conference on Neural Information Processing Systems*. 2011.

[59] David R Karger, Sewoong Oh, and Devavrat Shah. “Efficient crowdsourcing for multi-class labeling”. In: *ACM SIGMETRICS Performance Evaluation Review*. Volume 41. 1. ACM, 2013, pages 81–92.

[60] Gabriella Kazai. “In search of quality in crowdsourcing for search engine evaluation”. In: *European Conference on Information Retrieval*. Springer. 2011, pages 165–176.

[61] Gabriella Kazai, Jaap Kamps, and Natasa Milic-Frayling. “The face of quality in crowdsourcing relevance labels: Demographics, personality and labeling accuracy”. In: *ACM International Conference on Information and Knowledge Management*. ACM. 2012, pages 2583–2586.

[62] Faiza Khan Khattak and Ansaf Salleb-Aouissi. “Quality control of crowd labeling through expert evaluation”. In: *Proceedings of the NIPS 2nd Workshop on Computational Social Science and the Wisdom of Crowds*. Volume 2. 2011, page 5.

[63] Faiza Khan Khattak and Ansaf Salleb-Aouissi. “Robust crowd labeling using little expertise”. In: *International Conference on Discovery Science*. Springer. 2013, pages 94–109.

[64] Kickstarter. URL: `http://www.kickstarter.com/`.

[65] Ari Kobren et al. “Getting more for less: Optimized crowdsourcing with dynamic tasks and goals”. In: *Proceedings of the 24th international conference on world wide web*. International World Wide Web Conferences Steering Committee. 2015, pages 592–602.

[66] Klaus Krippendorff. *Content Analysis. An Introduction to its Methodology*. SAGE, 2012.

[67] Kumuluz. URL: `http://www.kumuluz.com/`.

[68] John Le et al. “Ensuring quality in crowdsourced search relevance evaluation: The effects of training question distribution”. In: *2010 workshop on crowdsourcing for search evaluation*. 2010, pages 21–26.

[69] John Le et al. “Ensuring quality in crowdsourced search relevance evaluation: The effects of training question distribution”. In: *SIGIR 2010 workshop on crowdsourcing for search evaluation*. Volume 2126. 2010.

[70] Quoc Le and Tomas Mikolov. “Distributed representations of sentences and documents”. In: *International Conference on Machine Learning*. 2014, pages 1188–1196.

[71] Matthew Lease. “On Quality Control and Machine Learning in Crowdsourcing”. In: *Human Computation*. Volume 11. 2011, page 11.

[72] Kyumin Lee, James Caverlee, and Steve Webb. “The social honeypot project: protecting online communities from spammers”. In: *Proceedings of the 19th international conference on World wide web*. ACM. 2010, pages 1139–1140.

[73] Sooyoung Lee, Sehwa Park, and Seog Park. “A quality enhancement of crowdsourcing based on quality evaluation and user-level task assignment framework”. In: *Big Data and Smart Computing (BIGCOMP), 2014 International Conference on*. IEEE. 2014, pages 60–65.

[74] Hongwei Li and Bin Yu. “Error rate bounds and iterative weighted majority voting for crowdsourcing”. In: *arXiv preprint arXiv:1411.4086*. 2014.

[75] Hongwei Li, Bin Yu, and Dengyong Zhou. “Error rate analysis of labeling by crowdsourcing”. In: *Machine Learning meets Crowdsourcing Workshop*. 2013.

[76] Hongwei Li, Bo Zhao, and Ariel Fuxman. “The Wisdom of Minority: Discovering and Targeting the Right Group of Workers for Crowdsourcing”. In: *World Wide Web*. Seoul, Korea, 2014, pages 165–176. ISBN: 978-1-4503-2744-2.

[77] Andrew Mao et al. “Volunteering versus work for pay: Incentives and tradeoffs in crowdsourcing”. In: *Human Computation*. 2013.

[78] Marketsandmarkets. *Edge Computing Market by Component (Hardware, Platform, Solutions), Application (Smart Cities, Location Services, Analytics, Augmented Reality), Organization Size (SME, Large Enterprises), Vertical, and Region - Global Forecast to 2022*. accessed 20-03-2018. October 2017. URL: https://www.marketsandmarkets.com/Market-Reports/edge-computing-market-133384090.html.

[79] Afra J. Mashhadi and Licia Capra. “Quality Control for Real-time Ubiquitous Crowdsourcing”. In: *UbiCrowd*. Beijing, China, 2011, pages 5–8. ISBN: 978-1-4503-0927-1.

[80] Panagiotis Mavridis, David Gross-Amblard, and Zoltán Miklós. “Using hierarchical skills for optimized task assignment in knowledge-intensive crowdsourcing”. In: *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee. 2016, pages 843–853.

[81] Tanushree Mitra, Clayton J Hutto, and Eric Gilbert. “Comparing person-and process-centric strategies for obtaining quality data on amazon mechanical turk”. In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM. 2015, pages 1345–1354.

[82] Kaixiang Mo, Erheng Zhong, and Qiang Yang. “Cross-task Crowdsourcing”. In: *SIGKDD*. Chicago, Illinois, USA, 2013, pages 677–685. ISBN: 978-1-4503-2174-7.

[83] Todd K Moon. “The expectation-maximization algorithm”. In: *IEEE Signal processing magazine*. Volume 13. 6. IEEE, 1996, pages 47–60.

[84] Robert Morris, Mira Dontcheva, and Elizabeth Gerber. “Priming for Better Performance in Microtask Crowdsourcing Environments”. In: *IEEE Internet Computing*. Volume 16. 5. Piscataway, NJ, USA, Sept. 2012, pages 13–19.

[85] Hayam Mousa et al. “PrivaSense: Privacy-Preserving and Reputation-Aware Mobile Participatory Sensing”. In: *Proceedings of the 14th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous)*. 2017.

[86] Derek G Murray et al. “The case for crowd computing”. In: *Proceedings of the second ACM SIGCOMM workshop on Networking, systems, and applications on mobile handhelds*. ACM. 2010, pages 39–44.

[87] Nasir Naveed et al. “Searching microblogs: coping with sparsity and document quality”. In: *Proceedings of the 20th ACM international conference on Information and knowledge management*. ACM. 2011, pages 183–188.

[88] Jerzy Neyman. “On the two different aspects of the representative method: the method of stratified sampling and the method of purposive selection”. In: *Journal of the Royal Statistical Society*. Volume 97. 4. JSTOR, 1934, pages 558–625.

[89] Quoc Viet Hung Nguyen et al. “Batc: a benchmark for aggregation techniques in crowdsourcing”. In: *International ACM SIGIR conference on research and development in information retrieval*. 2013, pages 1079–1080.

[90] *NLTK*. `https://www.nltk.org/`. Accessed: 2018-02-25.

[91] David Oleson et al. “Programmatic Gold: Targeted and Scalable Quality Assurance in Crowdsourcing.” In: *Human computation*. Volume 11. 11. 2011.

[92] Lawrence A Palinkas et al. “Purposeful sampling for qualitative data collection and analysis in mixed method implementation research”. In: *Administration and Policy in Mental Health and Mental Health Services Research*. Volume 42. 5. Springer, 2015, pages 533–544.

[93] Kalpana Parshotam. “Crowd computing: a literature review and definition”. In: *Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference*. ACM. 2013, pages 121–130.

[94] Alexander J Quinn and Benjamin B Bederson. “Human computation: a survey and taxonomy of a growing field”. In: *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM. 2011, pages 1403–1412.

[95] Quora. URL: `http://www.quora.com/`.

[96] Habibur Rahman et al. “Task assignment optimization in collaborative crowdsourcing”. In: *Data Mining (ICDM), 2015 IEEE International Conference on*. IEEE. 2015, pages 949–954.

[97] Leonard Andreevich Rastrigin. “Random Search in Optimization Problems for Multiparameter Systems”. In: 1967.

[98] Vikas C Raykar and Shipeng Yu. “Ranking annotators for crowdsourced labeling tasks”. In: *Advances in Neural Information Processing Systems 24*. Edited by J. Shawe-Taylor et al. Curran Associates, Inc., 2011, pages 1809–1817. URL: `http://papers.nips.cc/paper/4469-ranking-annotators-for-crowdsourced-labeling-tasks.pdf`.

[99] Vikas C Raykar and Shipeng Yu. “Eliminating spammers and ranking annotators for crowdsourced labeling tasks”. In: *Journal of Machine Learning Research*. Volume 13. Feb. 2012, pages 491–518.

[100] Vikas C. Raykar et al. “Supervised Learning from Multiple Experts: Whom to Trust when Everyone Lies a Bit”. In: *ICML*. Montreal, Quebec, Canada, 2009, pages 889–896. ISBN: 978-1-60558-516-1.

[101] Vikas C Raykar et al. “Supervised learning from multiple experts: whom to trust when everyone lies a bit”. In: *Proceedings of the 26th Annual international conference on machine learning*. ACM. 2009, pages 889–896.

[102] Vikas C. Raykar et al. “Learning From Crowds”. In: *J. Mach. Learn. Res.* Volume 11. 2010, pages 1297–1322.

[103] Reddit. URL: `http://www.reddit.com/`.

[104] Andrew Rosenberg and Julia Hirschberg. “V-measure: A conditional entropy-based external cluster evaluation measure”. In: *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*. 2007.

[105] Joel Ross et al. “Who are the crowdworkers?: shifting demographics in mechanical turk”. In: *CHI’10 extended abstracts on Human factors in computing systems*. ACM. 2010, pages 2863–2872.

[106] Peter J Rousseeuw and L Kaufman. *Finding groups in data*. Wiley Online Library Hoboken, 1990.

[107] Senjuti Basu Roy et al. “Task assignment optimization in knowledge-intensive crowdsourcing”. In: *The VLDB booktitle*. Volume 24. 4. Springer, 2015, pages 467–491.

[108] Jeffrey M. Rzeszotarski and Aniket Kittur. “Instrumenting the Crowd: Using Implicit Behavioral Measures to Predict Task Performance”. In: *UIST*. Santa Barbara, California, USA, 2011, pages 13–22. ISBN: 978-1-4503-0716-1.

[109] Mejdl Safran and Dunren Che. “Real-time recommendation algorithms for crowdsourcing systems”. In: *Applied Computing and Informatics*. 2016, pages –.

[110] Gerard Salton and Michael McGill. *Modern information retrieval*. 1983.

[111] Christina Sarasua et al. “Crowdsourcing and the Semantic Web: A research manifesto”. In: *Human Computation*. Volume 2. 1. 2015, pages 3–17.

[112] J Ben Schafer et al. “Collaborative filtering recommender systems”. In: *The adaptive web*. 2007, pages 291–324.

[113] Daniel Schall, Benjamin Satzger, and Harald Psaier. “Crowdsourcing tasks to social networks in BPEL4People”. In: *World Wide Web*. Volume 17. 1. Springer, 2014, pages 1–32.

[114] Daniel Schneider et al. “CSCWD: Five characters in search of crowds”. In: *Computer Supported Cooperative Work in Design (CSCWD), 2012 IEEE 16th International Conference on*. IEEE. 2012, pages 634–641.

[115] *Scikit*. `http://scikit-learn.org/stable/`. Accessed: 2018-02-25.

[116] Shelly Singh. *Blockchain Market worth 7,683.7 Million USD by 2022*. accessed 20-03-2018. 2017. URL: https://www.marketsandmarkets.com/PressReleases/blockchain-technology.asp.

[117] Rion Snow et al. "Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks". In: *Conference on empirical methods in natural language processing*. Association for Computational Linguistics. 2008, pages 254–263.

[118] Raymond T Sparrowe et al. "Social networks and the performance of individuals and groups". In: *Academy of management booktitle*. Volume 44. 2. Academy of Management Briarcliff Manor, NY 10510, 2001, pages 316–325.

[119] Thingiverse. URL: http://www.thingiverse.com/.

[120] Thingiverse. *Thingiverse - Digital Designs for Physical Object - About*. accessed 05-03-2018. URL: https://www.thingiverse.com/about/.

[121] Tian Tian and Jun Zhu. "Max-margin majority voting for learning from crowds". In: *Advances in Neural Information Processing Systems*. 2015, pages 1621–1629.

[122] Long Tran-Thanh et al. "Efficient Budget Allocation with Accuracy Guarantees for Crowdsourcing Classification Tasks". In: *AAMAS*. St. Paul, MN, USA, 2013, pages 901–908. ISBN: 978-1-4503-1993-5. URL: http://dl.acm.org/citation.cfm?id=2484920.2485063.

[123] Amazon Mechanical Turk. URL: http://www.mturk.com/.

[124] Twitter. URL: http://www.twitter.com/.

[125] Chris Van Pelt and Alex Sorokin. "Designing a scalable crowdsourcing platform". In: *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. ACM. 2012, pages 765–766.

[126] Luis Von Ahn and Laura Dabbish. "Labeling images with a computer game". In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM. 2004, pages 319–326.

[127] Luis Von Ahn et al. "recaptcha: Human-based character recognition via web security measures". In: *Science*. Volume 321. 5895. American Association for the Advancement of Science, 2008, pages 1465–1468.

[128] Jing Wang, Panagiotis G Ipeirotis, and Foster Provost. "Quality-based pricing for crowdsourced workers". In: 2013.

[129] Peter Welinder et al. "The multidimensional wisdom of crowds". In: *Advances in neural information processing systems*. 2010, pages 2424–2432.

[130] Jianshu Weng et al. "Twitterrank: finding topic-sensitive influential twitterers". In: *Proceedings of the third ACM international conference on Web search and data mining*. ACM. 2010, pages 261–270.

[131] Jacob Whitehill et al. "Whose Vote Should Count More: Optimal Integration of Labels from Labelers of Unknown Expertise". In: *Conference on Neural Information Processing Systems*. 2009, pages 2035–2043.

[132] Mark E Whiting et al. “Crowd guilds: Worker-led reputation and feedback on crowdsourcing platforms”. In: *arXiv preprint arXiv:1611.01572*. 2016.

[133] Wikipedia. URL: http://www.wikipedia.org/.

[134] Wikipedia. *Wikipedia:Contributing to Wikipedia*. accessed 05-03-2018. URL: https://en.wikipedia.org/wiki/Wikipedia:Contributing_to_Wikipedia.

[135] Mark D Wilkinson et al. “The FAIR Guiding Principles for scientific data management and stewardship”. In: *Scientific data*. Volume 3. Nature Publishing Group, 2016.

[136] Dejun Yang et al. “Incentive mechanisms for crowdsensing: Crowdsourcing with smartphones”. In: *IEEE/ACM Transactions on Networking (TON)*. Volume 24. 3. IEEE Press, 2016, pages 1732–1744.

[137] Kan Yang et al. “Security and privacy in mobile crowdsourcing networks: challenges and opportunities”. In: *IEEE communications magazine*. Volume 53. 8. IEEE, 2015, pages 75–81.

[138] Bin Ye, Yan Wang, and Ling Liu. “Crowd trust: A context-aware trust model for worker selection in crowdsourcing environments”. In: *2015 IEEE International Conference on Web Services (ICWS)*. IEEE. 2015, pages 121–128.

[139] Ming Yin and Yiling Chen. “Bonus or Not? Learn to Reward in Crowdsourcing.” In: *International Joint Conference on Artificial Intelligence*. 2015, pages 201–208.

[140] Man-Ching Yuen, Irwin King, and Kwong-Sak Leung. “TaskRec: A Task Recommendation Framework in Crowdsourcing Systems”. In: *Neural Processing Letters*. Volume 41. 2. 2015, pages 223–238.

[141] Zelda B Zabinsky. “Random search algorithms”. In: *Wiley Encyclopedia of Operations Research and Management Science*. Wiley Online Library, 2010.

[142] Omar F. Zaidan and Chris Callison-Burch. “Crowdsourcing Translation: Professional Quality from Non-professionals”. In: *HLT*. Portland, Oregon, 2011, pages 1220–1229. ISBN: 978-1-932432-87-9.

[143] W. Zhang, T. Yoshida, and X. Tang. “TFIDF, LSI and multi-word in information retrieval and text categorization”. In: *IEEE SMC*. 2008, pages 108–113. DOI: 10.1109/ICSMC.2008.4811259.

[144] Yuchen Zhang et al. “Spectral methods meet EM: A provably optimal algorithm for crowdsourcing”. In: *Advances in neural information processing systems*. 2014, pages 1260–1268.

[145] Shi Zhong. “Efficient online spherical k-means clustering”. In: *IJCNN*. Volume 5. 2005, 3180–3185 vol. 5. DOI: 10.1109/IJCNN.2005.1556436.

[146] Dengyong Zhou et al. “Aggregating ordinal labels from crowds by minimax conditional entropy”. In: *International conference on machine learning*. 2014, pages 262–270.

[147] Denny Zhou et al. “Learning from the wisdom of crowds by minimax entropy”. In: *Advances in neural information processing systems*. 2012, pages 2195–2203.