UNIVERSITÀ DEGLI STUDI DI MILANO

DIPARTIMENTO DI INFORMATICA

SCUOLA DI DOTTORATO IN INFORMATICA
XXVII CICLO

Consensus-based Crowdsourcing: Techniques and Applications

INF/01 INFORMATICA

Lorenzo Genta

Supervisor: Prof. Silvana Castano

Assistant supervisor: Dr. Stefano Montanelli

Headmaster of the Ph.D. School: Prof. Ernesto Damiani

Academic Year 2013/2014

Rien dans l'univers ne saurait résister
à un nombre suffisamment grand
d'intelligences groupées et organisées.
[Teilhard de Chardin]

Acknowledgements

My first acknowledgement goes to my advisor Prof. Silvana Castano and my co-advisor Dr. Stefano Montanelli, for their great support during my PhD course and, in particular, during this Thesis work. I would also like to thank Prof. Alfio Ferrara for his kind help and for his suggestions. During these three years I had the opportunity to work with them on many projects and I want to sincerely thank them for making me feel part of their research group.

My second acknowledgement goes to my friends for both their professional and personal support: Edoardo, Diego, Chiara, Anna, Matteo, Erik, Martina, Alberto, Eva, Ettore and many others. They gave me a great support in the last three years. I feel very lucky to have met such special persons.

Last but not least, I would like to thank my (extended) family. Alessandra, my sister, for her support and for her great advices. It is hard to admit, but I will miss our 6-years cohabitation. Cristina and Giorgio, my parents, for the freedom they left me in choosing my direction and for the great support they gave me on my decisions. Lianella, my grandmother, for always trusting and supporting me. Marco, Raffaella, and Nenè, my uncles, for their support, their suggestions, and for always helping me in reaching my objectives. I finally thank Luca for his help, his support, and for his suggestions. Thanks to him, these last three years have been the best time of my life.

Milano, February 9^{th}, 2015

Abstract

Crowdsourcing solutions are receiving more and more attention in the recent literature about social computing and distributed problem solving. In general terms, crowdsourcing can be considered as a social-computing model aimed at fostering the autonomous formation and emergence of the so-called *wisdom of the crowd*. Quality assessment is a crucial issue for the effectiveness of crowdsourcing systems, both for what concerns task and worker management. Another aspect to be considered in crowdsourcing systems is about the kind of contributions workers can make. Usually, crowdsourcing approaches rely only on tasks where workers have to decide among a predefined set of possible solutions. On the other hand, tasks leaving the workers a higher level of freedom in producing their answer (e.g., free-hand drawing) are more difficult to be managed and verified.

In the Thesis, we present the LiquidCrowd approach based on consensus and trustworthiness techniques for managing the execution of *collaborative tasks*. By collaborative task, we refer to a task for which a factual answer is not possible/appropriate, or a task whose result depends on the personal perception/point-of-view of the worker. We introduce the notion of *worker trustworthiness* to denote the worker "reliability", namely her/his capability to foster the successful completion of tasks. Furthermore, we improve the conventional score-based mechanism by introducing the notion of *award* that is a bonus provided to those workers that contribute to reach the consensus within groups. This way, groups with certain trustworthiness requirements can be composed on-demand, to deal with complex tasks, like for example tasks where consensus has not been reached during the first execution. In LiquidCrowd, we define a democratic mechanism based on the notion of *supermajority* to enable the flexible specification of the expected degree of agreement required for obtaining the consensus within a worker group. In LiquidCrowd, three task typologies are provided: *choice*, where the worker is asked to choose the answer among a list of predefined options; *range*, where the worker is asked to provide a free-numeric answer; *proposition*, where the worker is asked to provide a free text answer.

To evaluate the quality of the produced results obtained through LiquidCrowd consensus techniques, we perform a testing against the SQUARE crowdsourcing benchmark. Furthermore, to evaluate the capability of LiquidCrowd to effectively support a real problem, real case studies about web data classification have been selected.

Contents

1 Introduction **9**

2 Crowdsourcing systems **13**

2.1 Recruit/retain workers 14

2.1.1 Market-based solutions 16

2.1.2 Independent solutions 18

2.2 Contribution typology 19

2.2.1 Decide solutions 20

2.2.2 Create solutions 22

2.2.3 Mixed solutions 24

2.3 Contribution combination 26

2.3.1 Majority voting solutions 27

2.3.2 Enriched MV solutions 28

2.3.3 Statistical-based solutions 29

2.4 Worker evaluation 32

2.4.1 Gold-task solutions 32

2.4.2 Peer-review solutions 34

2.4.3 Statistical-based solutions 36

2.5 Comparative analysis and thesis contributions 38

3 The LiquidCrowd approach **44**

3.1 LiquidCrowd techniques for task management 47

3.2 LiquidCrowd techniques for worker management 51

3.3 LiquidCrowd techniques for candidate-answers identification 53

3.3.1 Choice task 54

3.3.2 Range task 55
3.3.3 Proposition task 59
3.4 Considerations about task typologies 61

4 The Argo crowdsourcing prototype 65
4.1 The Argo motivations and features 65
4.2 The Argo architecture 67
4.2.1 Task manager 67
4.2.2 Consensus manager 70
4.2.3 Reward manager 71
4.2.4 Uncommitment manager 72
4.3 Implementation issues 73
4.3.1 Spring-based implementation 74
4.3.2 Hibernate-based database access 75
4.4 Example of task execution 77

5 Evaluation of Argo against the SQUARE benchmark 80

6 Liquid Crowd application to web data classification and experimental evaluation 85
6.1 Presentation of the application cases 85
6.2 Web data classification: a feature based approach 86
6.2.1 The *in*Cloud construction process 88
6.3 LiquidCrowd for web data classification 89
6.3.1 Similarity evaluation 89
6.3.2 Cluster labelling 91
6.4 Experimental results and case-study-oriented evaluation 94
6.4.1 Similarity evaluation test case 95
6.4.2 Cluster labelling 98

7 Conclusions and future work 106
7.1 Future work 106

Bibliography 110

List of Figures

2.1 A classification framework for consensus-based crowdsourcing systems 15

3.1 The LiquidCrowd approach for task execution and quality assessment . 45
3.2 Candidate-answers identification in tasks with $t = range$ 56
3.3 The second-ballot management schema 62

4.1 The Argo architecture 68
4.2 Argo prototype - login page (a) and home page (b) 77
4.3 Argo prototype - example of choice task execution 78
4.4 Argo prototype - statistics page 79

6.1 An example of a portion of *in*Cloud 87
6.2 The *in*Cloud construction process 88
6.3 Examples of choice tasks 91
6.4 Examples of (a) range task and (b) proposition task 93
6.5 Examples of (a) range task execution and (b) proposition task execution 94
6.6 Relationship between workload and score for the case study sim-eval . 97
6.7 Relationship between workload and trustworthiness of workers for the case study sim-eval 98
6.8 Relation between workers workload, score, and trustworthiness for the case study sim-eval 99
6.9 Relationship between workload and score for the case study clu-lab . . 100
6.10 Relationship between workload and trustworthiness of workers for the case study clu-lab 101
6.11 Relation between workers workload, score, and trustworthiness for the case study clu-lab 102

6.12 Labelling results (1) 103
6.13 Labelling results (2) 104

Chapter 1

Introduction

Crowdsourcing solutions are receiving more and more attention in the recent literature about social computing and distributed problem solving [Yuen et al., 2011]. Initially, crowdsourcing became popular as an organizational model for online job planning and execution. Amazon Mechanical Turk (AMT) is a well-known example of this kind where a large number of workers (i.e., the crowd) is employed to perform massive bulks of work usually segmented in simple, repetitive tasks in exchange for a (money) revenue [Kittur et al., 2008]. More recent crowdsourcing applications have been proposed in many different contexts for execution of time-consuming activities where the use of automatic procedures is not completely effective or suitable, such as for example collaborative filtering [Starbird et al., 2012], reputation systems [Mashhadi and Capra, 2011], and web-resource tagging [Finin et al., 2010a]. In general terms, crowdsourcing can be considered as a social-computing model aimed at fostering the autonomous formation and emergence of the so-called *wisdom of the crowd* [Surowiecki, 2005].

In some cases, the role of CS is to involve workers in defining a reliable training set to use as input for active-learning algorithms [Yang et al., 2010]. In other cases, CS is invoked with the goal to validate/supervise the output of collaborative tasks, like for example deduplication and entity resolution [Whang et al., 2013], resource classification and labelling [Gomes et al., 2011], and entity matching/linking [McCann et al., 2008]. By collaborative task we refer to a task where a factual answer is not possible/appropriate or a task whose result depends on the personal point-of-view, knowledge, perception, expertise, and human understanding of the worker involved in

the task execution. Thus, the more workers agree on a certain answer, the more this answer is valid/accepted as a committed result for the task. In collaborative tasks, the crowd is thus decisive in generating an appropriate/convincing result.

Quality assessment is a crucial issue for the effectiveness of crowdsourcing systems, both for what concerns task and worker management. In particular, quality issues are related to questions like "how to evaluate when a task is successfully executed?", "how to avoid that poorly-accurate/malicious workers affect the overall quality of task results?". On one hand, a task is assigned to a single worker as a conventional solution and quality checking over the executed tasks is usually not enforced. The adoption of review mechanisms or agreement policies is sometimes proposed for task validation through the progressive involvement of additional workers in the task execution until a strongly-prevailing answer is provided [Barowy et al., 2012]. On the other hand, *score-based* mechanisms are generally employed for worker management, where the revenue is proportional to the number of executed tasks. The higher is the number of tasks, the higher is the worker revenue regardless the quality of the produced result. In the crowd, it is possible that some workers are "untrustworthy" in that they produce task results containing errors due to rush or insufficient skills. The introduction of incentives is a possible solution to stimulate workers in accurate task execution. However, at the best of our knowledge, the idea of calculating the worker revenue according to quality parameters is currently unsupported and it is only mentioned as a possible improvement [Barowy et al., 2012; Le et al., 2010]. The capability to evaluate/estimate the *worker trustworthiness* is a further solution for effective worker management in task assignment (e.g., a trustworthy worker can be employed for supervision and quality-checking of tasks executed by ordinary workers with undefined/low trustworthiness). Some solutions in this direction are recently being appearing [Demartini et al., 2012; Ipeirotis et al., 2010].

Another aspect to be considered in crowdsourcing systems is about the kind of contributions workers can make. Usually, crowdsourcing approaches rely only on tasks where workers have to decide among a predefined set of possible solutions [Finin et al., 2010b; Demartini et al., 2012; Barowy et al., 2012]. This happens for a twofold reason: on one hand, workers are often involved in improving the quality of results produced by automatic techniques. Thus, the tasks usually consists in a decision for the best

solution among a set of automatically-generated solutions. On the other hand, tasks leaving the workers a higher level of freedom in producing their answer (e.g., free-hand drawing) are more difficult to be managed and verified.

In the thesis, we present the LiquidCrowd approach for quality-assessing the results of crowdsourcing activities on collaborative tasks based on consensus and trustworthiness techniques. The main contributions of the thesis work and of LiquidCrowd are the following:

- *Use of trustworthiness to evaluate workers and their contributions*. In LiquidCrowd, we introduce the notion of *worker trustworthiness* to denote the worker "reliability", namely her/his capability to foster the successful completion of tasks. Furthermore, we improve the conventional score-based mechanism by introducing the notion of *award* that is a bonus provided to those workers that contribute to reach the consensus within groups. This way, groups with certain trustworthiness requirements can be composed on-demand, to deal with complex tasks, like for example tasks where consensus has not been reached during the first execution.

- *Use of supermajority-based consensus to evaluate task results*. In LiquidCrowd, each task is assigned for execution to a group of workers with a fixed number of participants. A task is considered as successfully completed if the group members reach the consensus on the task result/answer. To this end, we define a democratic mechanism based on the notion of *supermajority* to enable the flexible specification of the expected degree of agreement required for obtaining the consensus within a worker group.

- *Definition of a high-level approach valuable for different contribution typologies*. In LiquidCrowd, three task typologies are provided: *choice*, where the worker is asked to choose the answer among a list of predefined options; *range*, where the worker is asked to provide a free-numeric answer; *proposition*, where the worker is asked to provide a free-text answer. The proposed LiquidCrowd model supports all these three task typologies through a common supermajority-based approach.

To evaluate the quality of results obtained through LiquidCrowd consensus techniques, we perform a testing against the SQUARE crowdsourcing benchmark. Furthermore, to evaluate the capability of LiquidCrowd to effectively support a real problem, real case studies about web data classification have been selected. Web data classification is a typical domain of application of crowdsourcing, since the growing popularity of the Linked Data paradigm introduced a new way of exposing, sharing and connecting pieces of data, information and knowledge [Bizer et al., 2009]. In this context, a challenging issue is to provide techniques for effectively managing such a large scale of web data. A deserved solution is to aggregate linked data about a given user interest/target-topic according to their degree of similarity and to provide visual tools to enable interactive exploration modalities of similarity clusters for the user [Ferrara et al., 2014b; Bozzon et al., 2010]. In this context, we employed LiquidCrowd as a support/validation mechanism for assessing web data similarity and for labelling clusters of similar data. For the application of LiquidCrowd, we rely on the Argo prototype, the platform we developed for testing and experimentation of the proposed crowdsourcing techniques.

The thesis is organized as follows. Chapter 2 presents the state-of-the-art and a critical comparison of crowdsourcing systems. Chapter 3 presents the LiquidCrowd approach, its featuring components, as well as the LiquidCrowd techniques for consensus negotiation and trustworthiness calculation. In Chapter 4, the Argo prototype and related implementation choices are presented. The LiquidCrowd evaluation against the SQUARE benchmark is discussed in Chapter 5 and the application of LiquidCrowd to the web-data classification case studies is described in Chapter 6. Finally, concluding remarks are given in Chapter 7.

Chapter 2

Crowdsourcing systems

The term *crowdsourcing* is introduced for the first time in [Howe, 2006] where it is defined as an alternative solution to *outsourcing* (i.e., a company subcontracting some activity to a third-party) that involves an undefined (and generally large) network of people instead of well-known subjects. To better understand the issues related to crowdsourcing, the notions of *worker* and *requester* are introduced. The requester is an individual (e.g., a person, a company) who provides a task (or a set of tasks) to be solved. A worker is a user participating in the crowdsourcing activity offering his time/work in exchange for a reward (e.g., money, glory).

The most largely-accepted definition of crowdsourcing is given in [Arolas and de Guevara, 2012], where crowdsourcing is defined as *"a type of participative online activity in which an individual, [...], proposes to a group of individuals of varying knowledge, heterogeneity, and number, via a flexible open call, the voluntary undertaking of a task."*. A typical example of a crowdsourcing platform is Wikipedia[1], an online encyclopedia entirely built by the web community.

Thesis focus. In order to better position the thesis in the growing field of crowdsourcing systems, a first high-level distinction among *individual* and *collective* approaches has to be introduced [Geiger et al., 2011]. **Individual** approaches are those treating each gathered contribution in isolation (e.g., eBird [Sullivan et al., 2009]). In this kind of approach, the correctness of each contribution should be assessed by

[1] http://www.wikipedia.org

providing a set of constraint and verification rules that must be verified on a per-contribution basis. **Collective** approaches are characterized by searching for an emerging property that could only be obtained by combining together multiple contributions (e.g., AutoMan [Barowy et al., 2012]). In this kind of approach, gathered contributions can only be judged collectively or in relation to each other.

The LiquidCrowd approach is in the field of collective approaches where, for each single task, it is possible to gather a (high) number of contributions. Moreover, in LiquidCrowd there is no a-priori notion of *correct/good* result. In fact, we propose a consensus-based approach where the final task result is obtained through an analysis of the agreement among the workers. For this reason, correctness evaluation can only be performed by considering the contributions in relation with each other.

State-of-the-art classification. For the classification of the state-of-the-art on collective crowdsourcing approaches, we consider the four main issues defined in [Doan et al., 2011] and we refine/specialize them to provide a refined classification framework shown in Figure 2.1.

In the following sections, each first-level issue of Figure 2.1 is analysed and the current state-of-the-art solutions in the field of consensus-based crowdsourcing are presented. Finally, in Section 2.5, a comparative analysis of the most representative state-of-the-art approaches/solutions is presented, together with the original contributions of the thesis.

2.1 Recruit/retain workers

Definition. *Worker recruitment/retention* refers to the problem of hiring persons for the execution of crowdsourcing activities.

This issue is more related to crowdsourcing platforms than to crowdsourcing techniques. In fact, the matter is on how to design a crowdsourcing platform for enabling the involvement of new workers and for assigning them to the tasks to execute. An interesting analysis of this issue is presented in [Brabham, 2010]. In this work, the author performs a study on the members of the Threadless community looking for the main motivations for participating on the Threadless crowdsourcing activities [Lakhani and Kanji, 2008]. As a result of this study, the author notes the emergence of five main

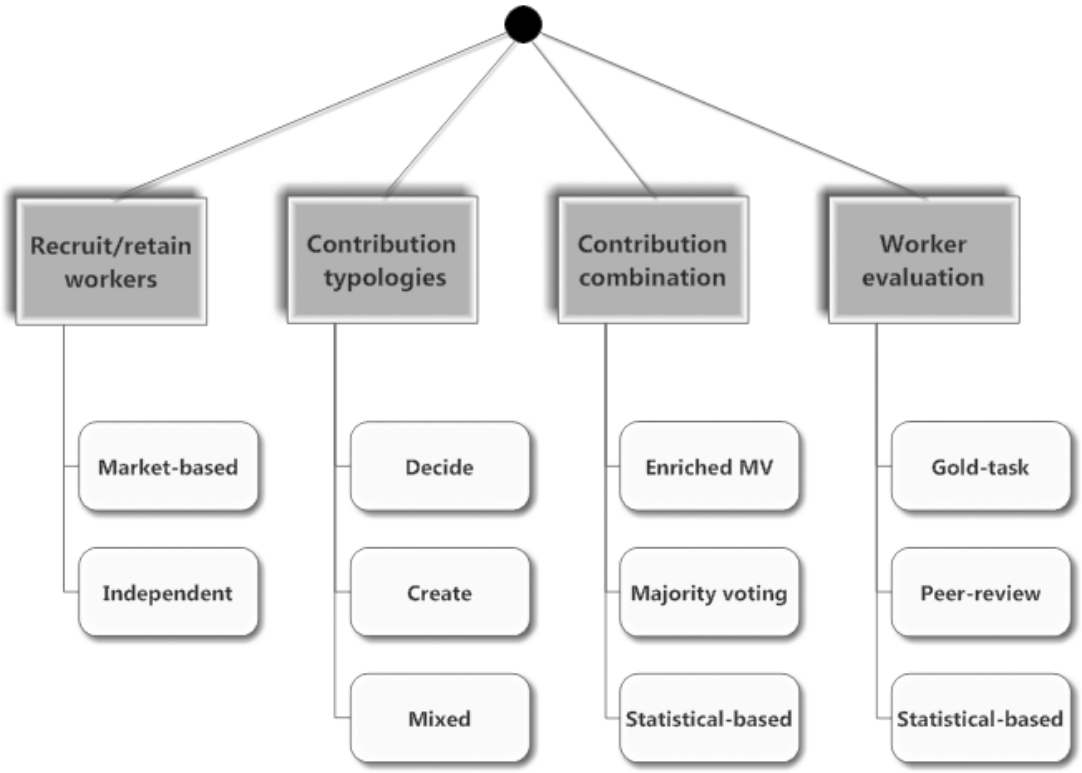


Figure 2.1: A classification framework for consensus-based crowdsourcing systems

themes: i) the opportunity to make money, ii) the opportunity to improve one's creative skills, iii) the opportunity for eventual freelance design work, iv) the love of community, and v) the addiction to the Threadless community.

We note that themes from i) to iii) are about worker recruitment (i.e., they involve opportunities for the workers who want to participate in the crowdsourcing activities), while themes iv) and v) are about worker retention (i.e., they involve the addiction of the workers to the crowdsourcing system).

About worker recruitment, the most widespread solution is to propose an open-access *task market* where workers are free to choose the task to work on, among a (usually large) set of available tasks. This way, workers are free to choose a task according to their creative skills and their money requirements. **Amazon Mechanical**

Turk (AMT)[2] is an interesting example of "task market", where people from all the world are allowed to take part to the crowdsourcing activity. In AMT, the workers are free to choose a task according to their preferences on the typology of the work and on the typology and amount of the proposed reward. Amazon claimed to have 100,000 workers subscribed to the AMT system in 2007. In a sense, this can be considered as a demonstration that the task-market model is an effective solution for worker recruitment [Kittur et al., 2008].

About worker retention, the crowdsourcing platform should present itself as a community, involving the workers in some addictive activity. In our opinion, worker-ranking mechanisms are a good solution for workers involvement, providing a competition-oriented community where workers are encouraged to improve their score/skills. **CrowdFlower**[3] represents an example of platform enforcing a worker scoring approach. Worker scoring is based on the quality of the worker contributions-history and is accessible to both workers and requesters.

Through an analysis of the state-of-the-art, two categories of solutions for worker recruitment/retention emerge:

- *Market-based*: solutions in this category rely on external crowdsourcing platforms (e.g. AMT, CrowdFlower). This way, the activities related to worker recruitment and retention are transferred to a commercial platform.
- *Independent*: solutions in this category perform worker recruitment relying on ad-hoc solutions or independent recruitment platforms.

2.1.1 Market-based solutions

The most widespread solution for market-based worker recruitment/retention is to rely on an online labour market service. This can be done in two ways: i) by directly relying on a labour market (e.g., AMT platform) or ii) by relying on intermediary platforms (e.g., CrowdFlower).

[2] http://www.mturk.com
[3] http://www.crowdflower.com

Finin et al. [2010b] In this work, a comparison among the AMT service and the CrowdFlower service is performed over a twitter-data labelling experimentation. The authors provides different tasks for AMT and CrowdFlower using a common collection of Twitter statuses and asking the workers to perform the same annotation work in order to fully understand advantages/disadvantages of both platforms. First of all, the authors stand that task generation is easier on AMT, due to usage of standard HTML/javascript markup, compared to the more complicated CML[4]. Secondly, the authors perform an analysis on quality improvement techniques for both platforms. On one hand, the AMT platform does not provide tools for gold-task-based worker evaluation, thus the authors implemented a Page-Rank-based algorithm to evaluate the level of agreement among each task. On the other hand, CrowdFlower provides gold-task-based worker evaluation, thus the authors relied on this functionality for worker evaluation.

Finally, the authors stand that both platforms are flexible, easy to use, capable of producing usable data, and very cost effective. Moreover, the authors stand that the additional features of CrowdFlower (i.e., gold-task-based worker evaluation) are very useful but they present some technical problems.

Jung and Lease [2011] In this work, a consensus-based approach for query/document relevance assessment is presented. The authors provide techniques for worker accuracy evaluation and the computed worker accuracy is used to weight the worker contributions. Worker recruitment/retention is based on the AMT system that is considered as a black-box relying on the following 2-steps procedure. In a first step, tasks are published on the AMT task market and worker contributions are gathered. In the second step, worker contributions are processed and no further interactions with the AMT system are required. This way, worker recruitment/retention is completely decoupled from the proposed crowdsourcing approach.

Sarasua et al. [2012] This paper presents CrowdMap, an ontology-alignment approach based on crowdsourcing techniques. The workflow of the proposed approach consists in: i) a task generation phase where in-doubt mappings are converted to crowdsourcing tasks, ii) a publishing phase, where the generated tasks are submitted to the crowd through the CrowdFlower APIs, and iii) a results analysis phase where the gathered contributions are analysed relying on the workers accuracy values provided by

[4] **C**rowdFlower **M**arkup **L**anguage

the CrowdFlower platform. In this work, the interaction with the crowdsourcing platform is continuous: the CrowdMap prototype provides mapping tasks and gold tasks to the CrowdFlower platform which performs not only worker recruitment/retention operations but also worker evaluation.

2.1.2 Independent solutions

Market based solutions have some limitations: i) workers can be rewarded only through a money payment ii) it is very difficult to select a group of people on the basis of their knowledge/reliability and, iii) task execution can only be triggered by the workers and there is no way to assign a task to a worker at a desired time. Some authors propose prototypes for independent crowdsourcing platforms to overcome these limitations.

Mashhadi and Capra [2011] In this paper, the authors analyse the problems connected with ubiquitous crowdsourcing (i.e., when the task of collecting information has to be performed continuously and in real-time, by an always changing crowd). More in detail, the proposed approach is in the context of mobility patterns analysis. The authors stand that state-of-the-art techniques for quality improvement fail in the considered context, thus, they propose novel techniques capable of improving data quality in real-time environments.

Due to the need for real-time task execution, a market-based approach is not suitable for the considered context. In fact, the authors also propose a prototype for a crowdsourcing platform presented as a smartphone application. Worker recruitment is performed manually, by selecting 100 people for the first test case. Worker retention is enforced through a game component, where workers are challenged to compete against each other to improve their score.

Kamar et al. [2012] The main aim of this work is to improve consensus-based techniques for galaxies tagging. The proposed approach is based on a Bayesian predictive model capable of reducing the required number of contributions per task by predicting the final result of a task. This work is based on a science project called *Galaxy Zoo*, born to enforce the power of the crowd to help experts in tagging galaxies with predefined features (e.g., galaxy typology, arms direction).

The presented work rely on the original worker contributions gathered from the

Galaxy Zoo project, which is adopted for worker recruitment and worker retention. In this project, worker recruitment and worker retention are obtained through an open access platform presented as a website, where workers are free to subscribe and take part to the crowdsourcing activities. The main motivations for workers participation are: i) love for the scientific progress and ii) the possibility to discover something new (i.e., glory).

Parameswaran et al. [2014] In this work, a crowdsourcing-based filtering approach is proposed, with the aim to extend a previous solution from the same authors called CrowdScreen [Parameswaran et al., 2012]. The focus of this work is on improving CrowdScreen by removing some simplifying assumptions and restrictions preserving, at the same time, the efficiency of the proposed approach.

Worker recruitment is performed by relying on students from a course on Human Computer Interaction of the Stanford University. The course relied entirely on evaluation by peer graders to judge the quality of the student submissions on a set of problems. Student exams and peer-reviews have been used as worker contributions for the proposed experimentation. We note that this publication represents a peculiar case of worker recruitment/retention, where no crowdsourcing platform is needed.

2.2 Contribution typology

Definition. *Contribution typology* refers to the problem of defining the kind of activities that workers can perform while participating to the crowdsourcing process.

This issue is related to the definition of possible interactions between workers and the crowdsourcing system. Existing solutions span from a simple multiple-choice selection [Whang et al., 2013] to a more complex free-hand picture drawing[5].

Referring to the work presented in [Malone et al., 2010], existing solutions can be distinguished in two macro-categories: *decide* and *create*. The presented distinction between decide and create approaches has been followed, for example, in [Minder and Bernstein, 2012], where a programming framework for engineering computation systems is proposed.

[5] http://www.tenthousandcents.com/

In the thesis, we introduce *mixed* as a third category, in order to represent crowdsourcing approaches where both create and decide contributions are supported.

2.2.1 Decide solutions

The solutions in the decide category are those where workers are asked to evaluate/select the most appropriate choice among a predefined set of alternatives (e.g., choose the most appropriate image for heading an article, decide whether to delete a Wikipedia article). Decide solutions can be distinguished into *individual* approaches and *group* approaches.

The individual approach is characterized by the fact that each worker can make his own decision independently by other workers. Referring to the work presented in [Malone et al., 2010], the individual decision can follow a *market* structure or a *social network* structure. In the first case, the individual decision is mediated by a formal exchange (e.g., money), as it occurs in online markets. Examples of this kind of approaches are Ebay[6], Threadless and iStockphoto[7]. In the second case, the members of the crowd form a network of relationships that is further exploited to obtain information on common interests, levels of trust and similarity of tastes. Examples of social network approaches are YouTube[8] "channels", Epinions.com[9] trust network and Amazon's[10] recommendation system.

The group approach is characterized by the fact that workers have to agree on the same decision. Group approach can be realized through the following strategies:

- *Voting*. This strategy is implied when each group member is able to choose between a collection of alternatives. The final result can be intended as the most voted alternative or as the top-k most voted alternatives. The voting operation can be explicit or implicit. In the first case the workers are directly asked to make

[6] www.ebay.com
[7] www.istockphoto.com
[8] www.youtube.com
[9] www.epinions.com
[10] www.amazon.com

a choice (e.g., the Digg[11] social bookmarking application), while in the second case the voting operation is hidden by a different worker action (e.g., YouTube ranks videos by the number of times have been viewed).

- *Averaging*. This strategy is implied when the problem to be solved by the crowd requires a free numeric answer. With an averaging strategy, the result of the group decisions is the average value of the worker contributions. An example of this strategy is presented in a first motivational work on crowdsourcing, where a group of workers was asked to estimate the average weight of an ox [Surowiecki, 2005].

- *Consensus*. When this strategy is implied, the contribution of a group of workers must be agreed by all (or by almost all) the group members. For a consensus strategy, the percentage of agreement (i.e., the number of workers agreeing on the same answer over the number of workers in the group) can be decided relying on different techniques. For example, in [Barowy et al., 2012] the percentage of agreement computed to minimize the probability that the workers obtained the consensus answering in a random way.

Raykar et al. [2010] In this paper, a probabilistic framework for supervised learning is presented. The proposed probabilistic model is able to learn annotator accuracy and true label jointly and measures the performance of each annotator in terms of *sensitivity* and *specificity* with respect to an unknown gold standard. Evaluated worker accuracy is further exploited to weight the worker contributions and prior knowledge about workers is also considered. The final estimation is performed by an Expectation-Maximization technique.

The proposed approach is presented for binary classification (i.e., the worker has to choose among two options) but an extension to the categorical classification problems is also introduced (i.e., the worker has to choose among multiple options).

Brew et al. [2010] In this work, a crowdsourcing approach for sentiment analysis and article classification is presented. The proposed approach is divided in two steps. In the first step, available articles are classified through an automatic classification system and proposed to the workers through thematic RSS channels. In the second

[11] www.digg.com

step, the workers evaluate the proposed articles with three possible interactions: i) positive, ii) negative, iii) not relevant. The first and the second options are used to express a sentiment on the considered article, while the third option is used to report an error in the classification step. All the collected contributions are finally exploited to retrain the automatic classification system on a daily basis. Finally, a novel metric for consensus evaluation is also introduced. The proposed metric considers the number of positive judgements related to the number of negative judgements and provides a value of the percentage of consensus on a specific task.

Tang and Lease [2011] In this work, a semi-supervised crowdsourcing approach based on both a MV technique and a statistical technique is proposed. The main idea behind this work is to rely on a Naive Bayes approach for estimating final results and worker accuracy values. The proposed approach is valuable for multiple-choice contributions. An experimentation is also provided, where workers are asked to evaluate the relevance of a web page to a structured user query. In this experimentation, possible answers for a task are four: Very relevant, Relevant, Not relevant, and I cannot view the content of the web page. The first three options are used to assess the relevance of the query for the considered web page, while the last option has been added to allow the worker to supply additional informations about the existence of the considered web page.

2.2.2 Create solutions

Solutions in the create category are those where workers are asked to generate something new (e.g., a piece of code, an image, a T-shirt design). Create solutions can be further distinguished into *collection* and *collaboration* approaches.

A collection approach is characterized by the fact that the crowdsourcing activities can be split into a number of atomic and independent tasks. The results produced by a collection approach can be used altogether or can be further selected. In the first case, after the parallel execution of the tasks, the results are finally put together to compose the overall crowdsourcing result (e.g., the *Ten thousands cents experiment*). In the second case, a *contest* phase can be executed in order to select the best result (or the top-k results) over all the provided answers (e.g., T-shirt selection on Threadless).

A collaboration approach is characterized by the fact that the execution of a single task requires to consider the elaboration of other tasks that are marked as dependant.

A collaboration approach is employed when the following conditions hold: i) the considered problem is not suitable for splitting in independent tasks, and ii) there are satisfactory ways of managing the dependencies between the individual pieces contributed by the members of the crowd. Examples of this kind of approaches are open source software (e.g., Linux O.S. project).

We note that, contribution combination techniques for the approaches in this category are more infrequent and, when provided, they are specifically-structured for the considered problem (i.e., they lack in generalizability).

Noronha et al. [2011] This paper introduces a crowdsourcing approach for nutrition analysis from food photographs. The presented approach is structured in three steps: *tag*, *identify*, and *measure*.

In the tag phase, workers are asked to draw boxes to isolate different foods visible in the proposed photograph. This step involves 2 initial workers. After the contribution collection, an algorithmic comparison is performed to assess the similarity of the proposed tags. If the tags are not similar enough, a second ballot phase is started, where three additional workers are asked to vote for the better tag.

The identify phase is further divided in two steps: in a first step, workers are asked to describe the tagged food using the natural language. In the second step, two workers are asked to choose among a predefined list of foods the entry that best matches the description proposed in the first step. Also in this phase, a second ballot is enforced if the contributions provided in the second step differ from each other.

In the measure phase, the workers are asked to estimate the portion size for each food matched in the identify phase. In this phase, no second ballot is provided: the worker contribution are averaged instead.

Starbird et al. [2012] In this paper, a crowd-behaviour study is proposed with the aim of identifying people tweeting from the ground during mass disruption events (i.e., events affecting a large number of people that causes disruption to normal social routines). In this crowdsourcing approach, the workers contributions are the tweets on twitter related to a specific event (i.e., free text contributions). A machine-learning approach is proposed where data extracted from tweets related to the Occupy Wall Street movement is analysed for feature extraction. Two main feature classes have been con-

sidered: *flat profile features*, based on the worker's profile, and *recommendation features*, based on how the worker interacted with other workers (e.g., retweets, follower growth). For generating the machine learning model, a Support Vector Machine has been employed.

Kuwabara and Ohta [2014] In this paper, an approach to construct a knowledge-base using crowdsourcing is proposed. The authors propose a crowdsourcing platform able to integrate human-based and machine-based services introducing the concept of *software agent*. The software agent acts as a human interface or as web service, depending on the used service. Moreover, an overview of the presented crowdsourcing platform is also proposed together with a high-level template of the work flow for revising task results. Finally, two motivating examples are introduced: the former is an application for natural language translation, and the latter is an application for content-creation in the e-learning environment. The considered contribution typology is a free-text answer but, due to the very high level of abstraction of the presented model, any typology of contribution can be adopted.

2.2.3 Mixed solutions

We note that, work proposing a mixed contribution typology are usually those introducing high-level crowdsourcing frameworks [Simperl et al., 2011; Minder and Bernstein, 2012; Bozzon et al., 2013]. Moreover, we note that these frameworks propose different contribution combination techniques for create and decide contributions. To the best of our knowledge, an approach/framework introducing a common contribution combination technique being valuable for mixed contribution typology has not yet been proposed.

Simperl et al. [2011] This paper presents a framework for crowdsourcing-based linked data management capable of integrating human and machine intelligence. The authors propose 5 task typologies for the interaction of human workers with the linked data cloud. The first typology is the *Identity resolution*, where workers are asked to choose if the proposed entities represent the same real-world concept or not. This task is used for the creation of SameAs links. The second typology is *Metadata Completion*, a create task where workers are asked to generate some missing data from scratch.

The third typology is *Classification*, where workers are asked to choose the best category to be associated to an entity. The fourth typology is *Ordering*, where workers are asked to choose a best-option among two proposals. Finally, the fifth typology is *Translation*, where workers are asked to translate a word or a sentence.

In this work, 3 task typologies are based on a decide contribution (i.e., identity resolution, classification, ordering), while 2 task typologies are based on a create contribution (i.e., metadata completion and translation).

Minder and Bernstein [2012] In this paper, *CrowdLang* is introduced as general-purpose framework and programmatic language for interweaving human and machine computation. The proposed programming language provides different task typologies as *interaction patterns*. The authors introduce *create* and *decide* interaction patterns. For each pattern, the framework defines a number of variations, strictly following the work presented in [Malone et al., 2010]. Finally, a step-by-step application example in the field of natural text translation is introduced.

The proposed approach does not provide specific techniques for contribution combination. It is to be intended as a general high-level framework whose purpose is to guide the generation of new crowdsourcing applications.

Bozzon et al. [2013] This work proposes a conceptual framework for modelling and controlling crowdsourcing computations. A crowdsourcing application is modelled as a composition of elementary task typologies. Nine elementary task typologies are proposed: five belonging to the decide category (i.e., choice, like, classify, order and group) and 4 belonging to the create category (i.e., score, tag, insert, modify). Based on these task typologies, different techniques are provided for result-quality assessment and spammers identification. For contribution combination, a simple MV approach with time-saving optimizations is proposed. Also in this framework, more advanced techniques for contribution combination are omitted and left for future work. Finally, 3 real application case studies are presented to assess the quality of the proposed approach.

2.3 Contribution combination

Definition. *Contribution combination* refers to the problem of assessing the final result for a task when multiple contributions have been gathered.

The need for contribution combination techniques depends on the proposed crowdsourcing approach. On one side, we have approaches where each task is assigned to a single worker and she/he provides her/his own task answer (individual approaches). In this case, worker contribution combination is straightforward (e.g., the ranking system of iStockphoto) or unnecessary (e.g., the Ten Thousand Cents experiment). On the other side, we have approaches where tasks are assigned to a group of workers and the task answer is the result of an agreement within the group (group approaches). In this case, this issue is strictly related to the issue presented in Section 2.2, given that the contributions combination technique depends on the contribution typology (e.g., a majority voting strategy can be adopted for a decide typology but is not directly applicable for a create typology).

In this respect, we note that individual approaches represent the conventional solution when the crowdsourcing approach requires to consider create contributions [Simperl et al., 2011; Franklin et al., 2011; Bozzon et al., 2013]. In our opinion, this means that there is a lack for a high-level combination technique being suitable for different mixed contribution typology. We also note that group approaches are frequently supported when a decide contribution typology is provided [Tang and Lease, 2011; Barowy et al., 2012; McCann et al., 2008]. Moreover, group solutions are usually context-dependant when the contribution typology leaves the worker a high level of freedom, such as for create contributions [Zaidan and Callison-Burch, 2011; Noronha et al., 2011].

In [Doan et al., 2011], the authors claim that the key problem related to worker contributions combination is on how to select the most appropriate worker contribution among the set of contributions provided by a group of workers rather than on combining them. In the current state-of-the-art, we found three main solutions for this problem: i) *majority voting*, ii) *enriched MV*, and iii) *statistical-based*.

2.3.1 Majority voting solutions

Majority voting (MV) techniques are the most widespread [Sheshadri and Lease, 2013] and are based on a common 2-step procedure. In a first step, all worker contributions for a task are collected, and identical contributions are grouped in identity classes. Each identity class is associated with a score corresponding to the number of grouped contributions. In a second step, the class with the higher score is selected to be the final task answer. Majority voting suffers on the tie-breaking problem: when two or more classes have equal score, further techniques are needed to break the tie and to select the final answer. Furthermore, MV relies on the fundamental assumption of workers acting independently and without knowing each-other.

Yang et al. [2010] The authors propose a crowdsourcing-based approach for extracting human labels for query-URL relevance evaluation. In this context, the task proposed to the workers shows a search-engine query and a list of URLs. The worker has to choose the relevance level of each URL for the given query among 5 option: *Perfect*, *Excellent*, *Good*, *Fair*, and *Bad*. One of the proposed contribution-combination techniques is based on a MV approach with an ad-hoc tie-breaking solution. At first, the most frequent labels are sorted in the order of most-relevant to least-relevant (i.e., from Perfect to Bad) and stored in a list. The majority vote is then picked as the label in the list which is indexed by $ceiling(m/2)$, where m is the number of most frequent labels.

Nowak and Rüger [2010] This paper is in the image-annotation field. The authors propose a comparison among labels produces by experts and crowdsourced labels. The crowdsourcing experiment is based on a MV approach where the agreement on a contribution cannot be worse to 50% (e.g., in a group of 10 workers, at least 5 workers agree on the same answer). The proposed agreement-threshold reduces the probability of ties and, probably for this reason, the tie-breaking problem is not considered. In the conclusions, the authors stand that the majority vote seems to filter some noise out of the annotations of the non-experts.

Nguyen et al. [2013] In this work, a crowdsourcing-based solution for how-to videos annotation is presented. The proposed approach is based on a three-steps procedure. In the first step, the workers have to identify the timestamps of the most important events

in the how-to video. In the second step, the workers have to label the timestamps found in the first step. In the third step, the workers have to select the best images representing the situation *before* and *after* the usage of a tool. The 1st and 3rd steps are executed relying on an individual approach, while 2nd step is executed both with an individual approach and with a group approach based on MV. The results demonstrate that MV significantly increases the accuracy of the produced labels.

2.3.2 Enriched MV solutions

Work proposing extensions on the Majority Voting approach are recently appearing. The main aim of such approaches is to improve the quality of the produced results by relying on a contribution-combination technique requiring a low computational cost. A first possible extension is given in [Tang and Lease, 2011], where a Majority Voting approach is used in conjunction with a Naive Bayes approach. Other work propose improvements to the MV approach by adding more restrictive rules for the extraction of the most-voted contribution.

McCann et al. [2008] In this work, a crowdsourcing-based schema-matching approach is proposed. Automatic matching tools usually produce *predictions* that are used in the matching procedure to improve and speed-up the matching process. In [McCann et al., 2008], the authors rely on a crowdsourcing approach to validate predictions and final results. The proposed technique, is based on a MV approach where a gap-criteria is introduced and non-fixed groups are considered. More in detail, given a question submitted to the crowd and three parameters min, max, gap, this approach consider the workers contributions as a continuous stream of answers (i.e., non-fixed group). The stream is interrupted when one of the following conditions hold: 1) the number of collected answers is greater than *min* and the gap between the most-voted answer and second most-voted answer is at least gap or 2) the number of collected answers is greater than *max*. Finally, the most-voted answer is returned.

The authors also demonstrate that the proposed mechanism produces right answers with high probability.

Tang and Lease [2011] In this work, a semi-supervised crowdsourcing approach based on both a MV technique and a statistical technique is proposed. The main idea

behind this work is to rely on a Naive Bayes approach for estimating final results and worker accuracy values. The authors note that the estimation of the model parameters can be simplified if all the true results are known a-priori. For this reason, the following 2-step procedure is proposed. In the first step, unknown results are computed through a MV technique. In the second step, the Naive Bayes approach is employed over the provided results in order to estimate the workers accuracies and the final results, iteratively. The proposed approach has also been compared against the simple MV technique and the Expectation-Maximization technique over two datasets. The results show a great improvement in accuracy for the proposed Naive Bayes technique with a small amount of supervision (i.e., expert-validated examples).

Barowy et al. [2012] This work presents a crowd-programming system called AutoMan, integrating human-based computations into a standard programming language. The proposed system also presents a quality control technique that can be considered as a more restrictive version of the MV approach. In AutoMan, the majority-voted result of a task is validated iff a predefined percentage of agreement is reached. The authors propose to compute this percentage with the aim to rule out the possibility that the results are due to random chance (with a desired level of confidence). When the required percentage of agreement is not reached, AutoMan automatically assigns the task to more workers, until the required agreement is reached or the maximum number of assignments has been performed.

2.3.3 Statistical-based solutions

Statistical-based techniques rely on probabilistic models where worker accuracy, task difficulty, correct answer and many other parameters are analysed through inference-based techniques. Most of the proposed approaches are based on Bayesian statistic techniques and present two main limitations related to i) the computational complexity and ii) the need for a bootstrapping phase. About i), the needed computational time varies depending on the adopted statistical-based technique but it is always greater than the time required by MV approaches. About ii), most of the analysed approaches need some training examples to produce high quality results. This means that, during the system bootstrap, the produced results will have low quality due to the incomplete parameters estimation. Moreover, like MV, statistical-based techniques usually

make simplifying assumptions of workers acting independently. We finally note that work proposing techniques capable to overcome these limitations are recently appearing [Venanzi et al., 2014; Feng et al., 2014] but their applicability to real scenarios has yet to be evaluated and confirmed.

Whitehill et al. [2009] In this work, a crowdsourcing approach for image-labelling called GLAD is proposed. The presented work is based on a probabilistic model where final results, workers accuracy, and tasks difficulty are estimated at the same time through an Expectation-Maximization approach. Worker contributions are limited to a boolean choice (i.e., the considered image belongs/not-belongs to a specific class) while worker accuracy and task difficulty are evaluated in $[-\infty, +\infty]$ and $[0, +\infty]$ intervals respectively. The authors also propose 2 empirical studies where GLAD is compared against MV demonstrating a significant gain in accuracy.

Demartini et al. [2012] In this paper, the authors propose an entity-linking approach (i.e., ZenCrowd) capable of identifying entities in natural language texts for linking to the Linked Open Data cloud. The presented work combines automatic techniques and crowdsourcing based techniques in order to i) improve the quality of produced results and ii) reduce the amount of work performed by the crowd. The provided approach can be divided in two steps. In the first step, state-of-the-art automatic techniques for entity extraction and matching are executed to find candidate links that will be analysed by a Decision Engine. The Decision Engine marks the results as *Excellent*, *Useless*, or *Uncertain*. In the second step, results marked as Uncertain are submitted to the crowd to be managed by human workers. The results provided by the crowd, together with the results produced by automatic techniques, are used as parameters in a probabilistic network. Through the proposed network, workers accuracy and final results are inferred relying on an Expectation-Maximization approach.

In this approach, worker contributions are limited to a boolean choice (i.e., the link is correct/incorrect) and the worker accuracy parameter can only be one of the following values $\{\text{Good}, \text{Bad}\}$.

Kamar et al. [2012] This work introduce a crowdsourcing approach (i.e., Galaxy Zoo) for galaxies classification. The authors propose a set of Bayesian predictive models and show how they can be used to combine human and machine contributions and

to predict workers behaviour. The proposed approach, models the problem of galaxy classification with 4 classes of features: *task features*, *vote features*, *worker features*, and *vote-worker* features. Task features represent the information related to the task while vote, worker and vote-worker features capture statistics about worker behaviour, worker participation and worker accuracy. The proposed Bayesian model, is able to predict the correct classification for a task, the next vote and the termination probability after the collection of different sets of contributions. Based on these prediction capabilities, a decision procedure is proposed in order to i) reduce the amount of work submitted to the crowd, and ii) improve the accuracy of the results. In future work, the authors propose to extend the model in order to include reasoning about timing and pricing of the crowdsourcing process.

Feng et al. [2014] In this paper, the INQUIRE framework is introduced. This framework is based on an incremental inference method based on two Bayesian models: the *question model* and the *worker model*. The question model is used to infer the final result of each question (i.e., task), while the worker model is used to evaluate the quality of the workers. Finally, the authors present two experiments and perform an analysis of the obtained results both in terms of accuracy and runtime. For the accuracy analysis, the proposed framework overcomes the previous statistical-based approaches and, consequently, the Majority Voting approaches. The runtime analysis shows that the INQUIRE framework achieves an execution time comparable to the time required by Majority Voting approach.

Venanzi et al. [2014] This work focuses on the ineffectiveness of the statistical-based approaches when only few labels per worker are available (i.e., the need for a bootstrapping phase). In this direction, the authors propose a novel community-based Bayesian aggregation model called *CommunityBCC* which assumes that crowd workers conform to a few different types. In order to achieve higher scalability, the model is split in 3 sub-models: object model, community model and worker model. The inference is performed locally on the sub-models and the results are finally merged by relying on standard message passing calculations. The provided empirical evaluation shows that the accuracy achieved by *CommunityBCC* is consistently higher with respect to the other state-of-the-art methods when the number of gathered labels is small.

2.4 Worker evaluation

Definition. *Worker evaluation* refers to the problem of assessing the accuracy of the workers and the quality of their contributions.

Workers and contributions evaluation is a very challenging task. On one hand, we have that manual evaluation performed by experts can be implied in small crowdsourcing experiments but is not suitable for real applications where hundred of thousands of contributions have to be analysed. On the other hand, solutions for automatic quality assessment of crowdsourcing results are recently being appearing. Three main categories of solutions can be distinguished: *gold-task*, *peer-review*, and *statistical-based*.

Another issue strictly concerned with worker evaluation is the *worker rewarding*. As far as we know, only few approaches directly consider the rewarding problem and in most cases only a basic solution is enforced. A worker is rewarded for a contribution iff the contribution is positively evaluated. Following the classification proposed in [Scekic et al., 2013], this basic approach is the most desirable rewarding mechanism and it is named *PPP* (pay-per-performance). The main limit is that when a worker contribution is erroneously marked as wrong, the worker will complain with the administrator for the missed reward (and/or for the consequent decrease of trustworthiness). This fact has also been discussed in [Barowy et al., 2012] and it is very difficult to address in large scale systems.

2.4.1 Gold-task solutions

In *gold-task*, the basic idea is to evaluate the worker trustworthiness by relying on entry/verification tests (i.e., gold tasks) whose correct answers are known in advance. This can be done in two ways. An option is to provide an *entry test* that a worker has to pass to become a trusted worker [Downs et al., 2010]. As another option, the gold questions are periodically provided to the worker in order to have a *continuous* trustworthiness verification [Le et al., 2010]. The main limit of gold-question solutions is that generating entry/verification tests is an expensive activity. Semi-automatic techniques for gold-question generation have also been proposed, but they present some limitations, too. For example, in [Oleson et al., 2011], solved tasks are used as gold

tasks for worker training. The main problem of this solution is that solved tasks could be wrong themselves. In this case, a worker has to complain when she/he believes that the gold question has a wrong answer, then the administrator has to manually supervise the gold question evaluation.
We also note that approaches for worker evaluation based on gold-tasks usually need domain-expert to validate the gold-task answers in order to evaluate the workers. This is a very expensive task, in particular if a periodic worker screening is needed.

Le et al. [2010] This paper propose a worker evaluation case study, based on entry tests and periodical tests. The main purpose of this work is to assess the quality improvements obtained by the usage of gold tasks in crowdsourcing activities. The authors performed 5 experiments on a dataset extracted from the internal search projects of a major online retailer where the workers are asked to assess the relevance of a product to a search query relying on 5 possible contributions (i.e., matching, not matching, spam, off-topic). Tests for worker-quality assessment, are performed on the CrowdFlower platform relying on previously-defined gold-tasks produced by expert users. The experiments are conducted by varying the distribution of answers in the training set from one skew to another one (e.g., in the first experiment the test set has a highly skewed distribution towards "Not matching"). The results show an accuracy improvement on "Not matching" tasks when the training set has a uniform distribution of correct answers. The authors finally point out the importance of using a well-structured training set for i) identify unethical workers and ii) train ethical workers more efficiently.

Downs et al. [2010] In this work, a screening process based on gold-tasks is introduced. The authors propose a gold-task model based on 2 questions that the worker has to answer after reading a small text. The two questions are different in their difficulty (i.e., the first one is an "easy question" while the second one is a "difficult question"). Each question presents a list of possible answers including i) a right answer and ii) a distractor (i.e., a wrong answer that seems true at a first sight). The obtained results have been analysed by grouping the crowd workers for age, profession and gender. An accurate analysis of the obtained results is finally presented.

Oleson et al. [2011] This work faces the problem of automatic gold-tasks generation. The authors propose an approach that manipulates already collected data (i.e., previously executed tasks) to generate new gold tasks based on the most common worker mistakes. The proposed approach is divided in four steps. In the first step, the worker mistakes are identified through manual audits. In the second step, a set of data mutations is defined. A data mutation, is a function that alters the structure of a task in order to produce a new task that a) differs from the original task enough to violate task requirements and b) looks similar to the original data. In the third step, the gold-tasks are collected: if the predefined set of gold tasks is too small, the proposed approach is to "bootstrap" the system with a set of easy-tasks requiring little gold to be completed. The results with higher worker agreement are finally used as new gold-tasks. The authors also consider the possibility that worker agreed on a wrong answer: to make the approach tolerant to these situations, the workers are able to contest the gold-task. When an automatically-generated gold-task has been contested by a high number of workers, it is automatically disabled. Finally, in the fourth step the target distribution of gold-tasks is defined. In this approach, the authors rely on a uniform distribution.

The proposed approach is also able to automatically generate error descriptions to be shown to the workers after the submission of an erroneous contribution.

2.4.2 Peer-review solutions

In *peer-review*, the basic idea is to manually review the solved tasks. The reviews are provided by the administrator or by other workers [Dow et al., 2011]. In the first case, the reliability of the reviews is increased, but the system scalability is very low, in that all the tasks have to be manually checked by a few supervisors. In the second case, the system scalability increases but the reliability of the reviews decreases. Another limit of peer-review solutions is the time consumption: in order to obtain a task answer the two-step procedure of execution and review has to be completed. We also note that, in peer-review systems, the attention is more focused on the reliability of workers contributions than on the reliability of workers themselves. This means that, in most work, peer-reviews are used to evaluate the quality of a contribution without modifying the perceived trustworthiness of the involved worker. Anyway, once the quality of the contribution has been assessed, the trustworthiness of the worker can be updated consequently relying on simple techniques (e.g., the trustworthiness can be computed

considering the ratio among the number of accepted contributions and the number of rejected contributions).

Bernstein et al. [2010] The main idea behind this work is to involve paid crowd workers for shorten, proofread, and edit parts of text documents. The authors propose a three-stages technical solution called *find-fix-verify*, involving peer-review for contribution evaluation.

In the find stage, the workers are asked to highlight parts of the text where an action is required (e.g., fix an error, shorten a sentence). A consensus technique is implied for answer validation where only patches where at least 20% of the workers agree are considered for the next stage.

In the fix stage, workers are asked to revise the text highlighted in the previous stage. Each worker propose his own revision and all the revision proposals are gathered and prepared for the verify stage.

The verify stage represent the peer-review based step. In this stage, workers are asked to vote the best option and to flag poor suggestions.

Finally, a real-case experimentation has been performed where 82% of grammar errors have been corrected and the text has been shortened to 85% of its original size.

Dow et al. [2011] In this work, an infrastructure for managing and providing feedback to crowd workers is presented. The authors critically analyse the most important features for feedback generation (called dimensions of the design space). The considered issues includes *timeliness* (i.e., when should feedback be shown), *specificity* (i.e., how detailed should feedback be), and *source* (i.e., who should provide feedback). Finally, the *Shepherd* prototype, an infrastructure for managing and providing feedback to crowd workers, is introduced. The main aim of this work is to provide an high-level administrative tool for graphic-based work progress visualization that could be integrated in future crowdsourcing platforms. Shepherd generates a timeline view base on a Gantt chart, showing when workers accept a task, how long they work on the task and how many tasks a worker complete within a batch. Through this view, requesters can monitor incoming work and provide feedback using specially designed forms. In future work, the authors will provide tools for generating feedback templates that will enable peer-review among workers.

Hansen et al. [2013] This paper presents technique for peer-reviewing in the field of document transcription (i.e., document digitalization). Three quality control mechanisms are presented. The first one, called *A-B-ARB* is based on the principle of *arbitration*: two crowd workers A and B transcribe a document independently and the discrepancies are passed to a third arbitrator (ARB) that makes the final decision. The arbitrator is also free to correct the contributions of A and B. The second mechanism is called *A-R* and consists in a worker A transcribing a document and a reviewer R that has to identify and correct possible errors produced by A. The third mechanism, called *A-R-RARB*, adds the RARB check to the A-R mechanism. The worker A transcribes a page, the worker R reviews the page and each discrepancy is reviewed by a third arbitrator (RARB).

The authors also present some experimentations and introduce two main findings: i) the peer-review processes (A-R and A-R-RARB) are a lot faster than the arbitration mechanism but the arbitration mechanism achieve higher accuracy and ii) arbitration of the peer-review edit (i.e., A-R-RARB) does not increase the result quality, suggesting that the simple A-R mechanism is preferable in the considered context.

2.4.3 Statistical-based solutions

In *statistical-based* approaches, the basic idea is to simultaneously infer the worker accuracy and the right answer by relying on statistical techniques. For example, in [Ipeirotis et al., 2010], the authors propose an Expectation-Maximization technique capable of separating the unrecoverable error rate from bias. These techniques also present some limitations. On one hand, high number of worker contributions is needed in order to be able to assess the worker trustworthiness. This means that workers will not know if their contributions have been accepted or not until a (generally long) period of time is expired. The worker trustworthiness is not evaluated when she/he contributes to the crowdsourcing activities with only a few task executions (otherwise the evaluation would be inaccurate). On the other hand, these techniques have a high computational cost. For these two reasons, it is difficult to apply techniques based on pure statistical inference for large-scale crowdsourcing projects.

Dawid and Skene [1979] This work introduces for the first time the idea of relying on an Expectation-Maximization approach for estimating interest parameters related to observers errors. This work was intended for application to the medical field, where the compilation of patient records is often subject to errors of measurement. The transportation of this approach to the crowdsourcing field can be done by interpreting the compiler of the patient record as the worker and the patient-record compilation as the performed task.

In this paper, two cases are considered: when true answers are available and when true answers are not available. The latter is the most interesting for the crowdsourcing field and is based on the following four steps procedure: i) obtain an initial estimate of the missing data (e.g., righ answer), ii) calculate the Maximum-Likelihood estimates for the needed parameters (e.g., worker trustworthiness, task difficulty), iii) re-compute new estimates for missing data, and iv) repeat steps ii) and iii) until convergence of the Maximum-Likelihood estimates and the missing data estimates.

Joglekar et al. [2013] In this work, statistical techniques for worker error rate and worker accuracy estimation are presented. The basic model relies on three parameters for each worker: the error probability, the confidence interval, and the confidence level. The considered contribution typology is a baseline "Yes/No" decide typology.

In order to assess worker error probability, two scenarios are presented: the *3-differences scheme* and the *general differences scheme*. The former scenario models a situation where 3 workers (let say A, B, and C) are involved with an unknown error probability. In this scheme, the worker error probability is estimated by observing worker pair disagreement rate (e.g., for worker A, his/her agreement with worker B and with worker C is considered). The latter scenario is an extension of the model introduced in the former scenario for more than 3 workers. In this case, the error estimation for a worker A is done by dividing all the remaining workers in two disjoint sets S and T. Each set is treated as a *super-worker* whose answer is the majority answer of the corresponding set (e.g., the set S is associated with a super-worker whose answer is the majority answer of set S). Finally, the 3-difference scheme is applied to worker A and super-workers T and S. Finally, experimental results for the two proposed schemes are presented.

Matsui et al. [2014] This work introduces a statistical-based crowdsourcing approach for items ordering. The authors propose a statistical model of the generative process of worker responses model and apply an Expectation-Maximization approach to obtain estimates for the true rank vectors as well as the worker ability parameters. The proposed approach relies on a distance-based model for orders extended with a crowdsourcing-related *concentration* parameter, representing the worker accuracy. The estimation approach is based on two steps: in the first step, the true rank vectors are inferred while in the second step, workers accuracy parameters are optimized.

An experimentation of the proposed approach is also presented, based on two datasets: word ordering and sentence ordering. The authors present the obtained results in terms of error rate and propose a comparison with a baseline crowdsourcing approach.

2.5 Comparative analysis and thesis contributions

In Table 2.1, the main features of solutions/approaches surveyed in the previous sections are summarized and critically compared. Main considerations and open issues are discussed in the following.

Workers evaluation. Two considerations emerge by analysing the worker evaluation techniques adopted in the state-of-the-art publications. The first consideration is about worker trustworthiness assessment. As far as we know, none of the currently available solutions is capable to address the limits introduced in Section 2.4. The second consideration is about worker rewarding. We note that worker rewarding is strictly connected with worker evaluation, in that, if the crowdsourcing approach is able to perform an evaluation of the worker trustworthiness, it will be consequently possible to perform selective rewarding (according to the calculated trustworthiness). In Table 2.1, we identified two possible kinds of worker rewarding: *non-selective rewarding* (NR), and *selective rewarding* (SR). The former is enforced when the worker rewarding technique is independent from the quality of the contributions (i.e., the workers are rewarded for each contribution). The latter is enforced when the worker rewarding technique is capable to distinguish among valuable and non-valuable workers. We also note that only a few approaches propose techniques for worker rewarding and, in most of the cases, a non-selective approach is enforced. Only two publications take into account

	Worker recruitment/retentention	Contribution typology	Contribution combination	Worker evaluation
Finin et al. [2010b]	Market-based	Decide	Majority Voting	Gold task (NR)
Jung and Lease [2011]	Market-based	Decide	Majority Voting (WV)	Gold task
Sarasua et al. [2012]	Market-based	Decide	Majority Voting	-
Mashhadi and Capra [2011]	Independent	Decide	Statistical-based (WV)	Statistical-based
Yang et al. [2010]	Independent	Decide	Majority Voting	-
Nowak and Rüger [2010]	Market-based	Decide	Majority Voting	- (NR)
Nguyen et al. [2013]	Market-based	Decide	Majority Voting	- (NR)
Whitehill et al. [2009]	Market-based	Decide	Statistical-based	Statistical-based
Demartini et al. [2012]	Market-based	Decide	Statistical-based (WV)	Statistical-based
Kamar et al. [2012]	Independent	Decide	Statistical-based (UM)	Statistical-based (NR)
McCann et al. [2008]	Independent	Decide	Enriched MV	Gold task
Tang and Lease [2011]	Market-based	Decide	Enriched MV	Statistical-based
Barowy et al. [2012]	Market-based	Decide	Enriched MV	- (SR)
Raykar et al. [2010]	Market-based	Decide	Statistical-based (WV)	Statistical-based
Brew et al. [2010]	Independent	Decide	Enriched MV	-
Parameswaran et al. [2014]	Independent	Decide	Statistical-based (WV)	Statistical-based
Noronha et al. [2011]	Market-based	Create	Enriched MV	-
Starbird et al. [2012]	Independent	Create	Statistical-based	-
Kuwabara and Ohta [2014]	-	Create	-	- (NR)

Simperl et al. [2011]	Market-based	Mixed	Majority Voting	-
Minder and Bernstein [2012]	Independent	Mixed	-	-
Bozzon et al. [2013]	Independent	Mixed	Majority Voting (UM)	Gold task (SR)
Le et al. [2010]	Market-based	Decide	Majority Voting	Gold task
Downs et al. [2010]	Market-based	Decide	-	Gold task
Oleson et al. [2011]	Independent	Decide	-	Gold task
Bernstein et al. [2010]	Market-based	Create	Enriched MV	Peer-review (NR)
Dow et al. [2011]	Market-based	-	-	Peer-review
Hansen et al. [2013]	Independent	Create	-	Peer-review
Dawid and Skene [1979]	-	Decide	Statistical-based	Statistical-based
Matsui et al. [2014]	Market-based	Decide	Statistical-based	Statistical-based (NR)
Joglekar et al. [2013]	Independent	Decide	Statistical-based (WV)	Statistical-based
Feng et al. [2014]	Market-based	Decide	Statistical-based (WV)	Statistical-based
Venanzi et al. [2014]	-	Decide	Statistical-based (WV)	Statistical-based

Legend:

(WV) - uses a weighted voting strategy

(UM) - uncommitment management is provided

(NR) - worker rewarding is considered (for any contribution)

(SR) - selective worker rewarding is considered (only for accepted contributions)

Table 2.1: Comparative analysis

selective rewarding techniques [Bozzon et al., 2013; Barowy et al., 2012]. In our opinion, there is a lack in the state-of-the-art for worker rewarding techniques capable of distinguishing worker efforts and contribution quality.

Thesis contribution. We propose a worker trustworthiness assessment approach based on the ability of the worker to be part of the consensus. With respect to peer-review solutions, LiquidCrowd does not perform an explicit review step. The review of each task is performed through consensus analysis and it is an essential part of the proposed approach. With respect to gold-task solutions, LiquidCrowd does not require any manual supervision activity. With respect to statistical-based solutions, the proposed approach presents no need for supervision. Furthermore, the score mechanism of LiquidCrowd based on a fixed salary and on a variable award provides a fair solution for worker revenue achieving the following two results: 1) it distinguishes worker effort (i.e., the work time) and contribution quality (i.e., the work quality) and 2) it reduces complaints due to unsatisfaction for unpaid task that are not committed. We also note that the proposed selective-rewarding mechanism provides a valuable solution to the problem of workers producing high amounts of poorly-qualitative contributions presented in [Mason and Watts, 2010]. In fact, in LiquidCrowd the award is only given to the workers producing results that are in agreement with the majority of the members of the group: this way, workers are driven to keep high the quality of their contributions.

Contribution combination. About this issue, we note that only few approaches have explored possible extensions to the majority voting technique. Moreover, such approaches often rely on fixed majority-thresholds, meaning that low flexibility is enforced. A possible improvement is to enforce the capability to dynamically set the majority-thresholds according to the considered case study. As far as we know, no work currently supports such a kind of flexibility. A possible extension to the majority-voting technique is the application of a *weighted-voting* strategy (WV), where worker contributions are weighted based on the worker trustworthiness. The weight of a contribution is usually proportional to the trustworthiness of the worker who provided it. This way, workers with high trustworthiness will have more importance in the consensus verification. The application of a weighted voting strategy is very frequent in statistical-based approaches but has been proposed for improvement of majority-voting techniques only in [Jung and Lease, 2011]. Another possible extension to the major-

ity voting technique could be to introduce solutions for *uncommitment management* (UM). For example, uncommitted tasks could be re-assigned to more accurate workers in order to increase the probability of a second-try commitment. As far as we know, uncommitment management techniques have been explicitly considered only in [Kamar et al., 2012], where a task-termination prediction technique is provided in order to avoid further task assignments when a low commitment probability is computed. Anyway, even in [Kamar et al., 2012], task-reassignment techniques techniques have not been proposed.

Thesis contribution. The LiquidCrowd approach relies on the notion of *group* of workers to automatically supervise the contributions of each single independent worker involved in the execution of a task. The main improvement with respect to the majority voting technique is the introduction of the *bop-constraint*, ensuring a majority that can not be controlled by a single worker. Moreover, the answer to a committed task satisfies the supermajority mechanism thus ensuring a large consensus among the involved workers. The enforcement of a weighted voting strategy is proposed to exploit the information we have about worker trustworthiness and the enforcement of groups with a prefixed size ensures that the committed task answer is the result of a certain worker effort that is known in advance including possible re-executions. As a difference with [Barowy et al., 2012], in LiquidCrowd it is not possible that tasks are committed over groups with a growing size, where the worker effort is not known in advance. As another difference with [Barowy et al., 2012], we note that LiquidCrowd adopts a more reliable solution where an explicitly uncommitted task is assigned to another group of workers with higher trustworthiness values and the answers collected in the first execution are discarded to avoid that wrong answers in uncommitted executions can negatively affect the subsequent executions. Moreover, the maximum number of task executions prefixed in LiquidCrowd ensures the task termination, regardless consensus verification.

Contribution typology. In Table 2.1, we note that only few solutions are proposed that support decide and create contributions at the same time [Simperl et al., 2011; Minder and Bernstein, 2012; Bozzon et al., 2013]. However, in these cases, techniques for contribution combination and worker evaluation are sometime left as a future work and sometimes are only suitable for decide contribution typology. Moreover, we note that the techniques presented in [Bozzon et al., 2013] are not applicable to create typology

in a straightforward manner. As such, we observe that a limit that still characterizes existing crowdsourcing systems is related to the capability to provide a single, comprehensive suite of techniques that manage combination and evaluation applied to mixed contribution typology.

Thesis contribution. In LiquidCrowd, *decide* and *create* contribution typologies are enforced within a unique crowdsourcing approach. The proposed approach provides high level techniques for managing contributions combination, worker trustworthiness assessment and worker rewarding relying on a common supermajority-based technique.

Worker recruitment/retention. About this point, we observe that market-based solutions are more popularly adopted than independent systems. However, independent systems are the preferred solution when there is the need to enforce crowdsourcing techniques that present peculiar/innovative aspects (e.g., novel rewarding mechanisms, ad-hoc task scheduling). This is due to the fact that it is difficult to embed novel crowdsourcing mechanisms into a prefixed crowdsourcing market. In this cases, the development of an independent crowdsourcing platform is easier and more effective than adapting/creating interfaces to an existing labour market.

Thesis contribution. We developed the Argo prototype, a crowdsourcing platform enabling the customization of the crowdsourcing process under different points of views (e.g., worker evaluation, contribution combination, contribution typology).

Chapter 3

The LiquidCrowd approach

Consider a problem to manage through crowdsourcing. The problem is split in a (usually large) corpus of atomic tasks to be performed and a crowd of workers is available to participate to the crowdsourcing activities. Figure 3.1 shows the LiquidCrowd approach, where the role of consensus and trustworthiness in task and worker management are evidenced. As it occurs in conventional approaches, LiquidCrowd follows a basic crowdsourcing workflow where i) a task is assigned to workers, ii) the task is scheduled for execution, and iii) the involved workers are paid for their effort. With respect to such a kind of basic workflow, LiquidCrowd is characterized by three main distinguishing features.

Group-oriented task assignment. In LiquidCrowd, a task is assigned to a group of workers instead of a single worker. A group of workers $\mathcal{G} = \{\mathcal{W}_1, \ldots, \mathcal{W}_{s_\mathcal{G}}\}$ is a collection of workers where each member $\mathcal{W} \in \mathcal{G}$ autonomously executes the received task and independently produces the answer according to her/his personal problem-understanding and skill. The number of members of $\mathcal{G}$ is called *size of* $\mathcal{G}$ and it is denoted $s_\mathcal{G}$. In LiquidCrowd, groups are *hidden* and *dynamic*, meaning that i) the members of a group $\mathcal{G}$ are not aware of being involved in $\mathcal{G}$ for the execution of a certain task, and ii) the composition of a group $\mathcal{G}$ changes from one task to another. This is done to avoid mutual and history-based influence among workers on the answer to be provided for a task.

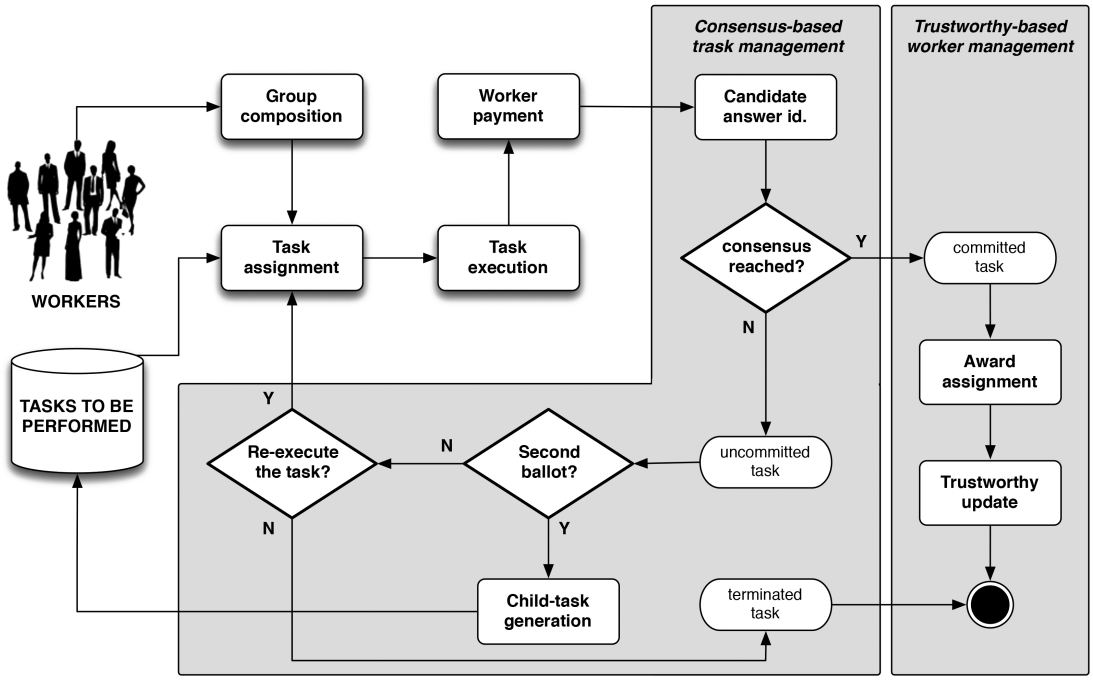


Figure 3.1: The LiquidCrowd approach for task execution and quality assessment

Consensus-oriented task evaluation. In LiquidCrowd, the result of a task depends on the level of agreement (i.e., consensus) among the group workers involved in the task execution. A peculiar feature of LiquidCrowd is that a task can be scheduled for re-execution when the expected level of agreement is not satisfied in the current execution. According to this, a task $\mathcal{T} = \langle P, S, E, \bar{A} \rangle$ is an atomic unit of work where P is the parent task, S is the task structure, E is the task execution history, and $\bar{A}$ is the committed answer selected as result of the task $\mathcal{T}$. The parent task P is a task $\mathcal{T}_p$ whom execution determined the structure of the task $\mathcal{T}$. From a different point of view, the execution of task $\mathcal{T}$ is the continuation of the last execution of the task $\mathcal{T}_p$ (more details on the parent task will be provided in Section 3.1). The task structure $S = \langle d, c, t, PA \rangle$ is composed by a description d, a context c, a typology t, and a set of possible answers PA. The description d is a short text synthesizing the task goal, while the context c is a short text providing the additional information (if required) to consider in performing the task. The task typology $t \in \{choice, range, proposition\}$ describes the kind of task:

- *Choice* is a task where the worker has to select the preferred option among a set

of pre-defined alternatives;

- *Range* is a task where the worker has to choose the preferred value in a given range of possible values;
- *Proposition* is a task where the worker has to formulate a free-text answer to the task question.

Finally, the structure of a task contains the set of possible answers *PA*. In particular, *PA* contains a set of predefined, alternative options when $t = choice$ and a range of values when $t = range$. The set *PA* is empty when $t = proposition$ since no initial answer possibilities are shown to the user.
The execution history $E = [e_1, \ldots, e_k]$ contains the list of executions of the task $\mathcal{T}$. A task can be executed up to a maximum number k of times, depending on the outcome of the consensus verification step. A task execution $e \in E$ is defined as $e = \langle \mathcal{G}, \mathcal{A}^*, q, th_\tau, s_e \rangle$ where $\mathcal{G}$ is the group that received the task $\mathcal{T}$ for execution, and $\mathcal{A}^* = \{A^1, \ldots, A^{s_\mathcal{G}}\}$ is the set of answers provided by the members of $\mathcal{G}$ in response to $\mathcal{T}$. The gathered answers $\mathcal{A}^*$ are processed in the *candidate-answer identification* step. A candidate-answer $CA = \langle V_{CA}, \mathcal{G}_{CA} \rangle$ is defined as an equivalence class over the worker answers, where V_{CA} is the representative value of the equivalence class and $\mathcal{G}_{CA} \subseteq \mathcal{G}$ is the support group of V, namely the set of workers that submitted an answer $A \in CA$. Finally, q is the *quorum*, th_τ is the *trustworthiness threshold*, and s_e is the final status of the execution. The quorum $q \in (0.5, 1]$ is the minimum level of consensus required within the group $\mathcal{G}$ for considering the task $\mathcal{T}$ as successfully completed. The trustworthiness threshold $th_\tau \in [0, 1]$ is the minimum level of trustworthiness that a worker needs to exhibit for being included in the group $\mathcal{G}$, and thus for being involved in the execution of the task $\mathcal{T}$. The execution status $s_e \in \{\mathsf{committed}, \mathsf{uncommitted}, \mathsf{terminated}\}$ expresses the outcome of task execution and it is decided on the result of consensus verification over the set of answers $\mathcal{A}^*$ provided by the $\mathcal{G}$ members as follows:

- Committed: it is selected when the consensus over the collected task answers $\mathcal{A}^*$ is obtained. In this case, the task is considered as successfully completed and the task result $\bar{A}$ is returned.
- Uncommitted: it is selected when the consensus over the collected task answers $\mathcal{A}^*$ is not obtained. In this case, the task needs to be re-executed. The task

assignment step is then invoked to prepare a new execution of the task $\mathcal{T}$ and to compose a new group $\mathcal{G}'$ with different workers for assignment of the task to re-execute.

- Terminated: it is selected when the consensus over the collected task answers $\mathcal{A}^*$ is not obtained and the maximum number k of task executions is reached. A terminated task has no result (i.e., $\bar{A} = NULL$) and it is considered as unsuccessfully completed. The task termination is a further peculiar feature of LiquidCrowd to avoid an infinite loop of re-assignments for those "controversial" tasks in which workers are not able to reach the consensus.

Trustworthiness-oriented worker evaluation. In LiquidCrowd, the worker evaluation aims at keeping into account not only the mere effort spent in executing tasks, but also the quality of the effort provided. A worker $\mathcal{W}$ is characterized by a *worker score* $\sigma_{\mathcal{W}}$, a *worker trustworthiness* $\tau_{\mathcal{W}}$, and a *worker status* $s_{\mathcal{W}}$. The worker score $\sigma_{\mathcal{W}}$ represents the worker revenue composed by i) a *salary*, the payment the worker receives each time she/he executes a task, regardless the consensus verification, and ii) an *award*, a bonus the worker receives each time she/he contributes to commit the task. The worker trustworthiness $\tau_{\mathcal{W}}$ represents the ability of $\mathcal{W}$ of being awarded based on the task execution history. Finally, the worker status $s_{\mathcal{W}} = \{available, unavailable\}$ describes the worker willingness to be inserted in a group for executing tasks.

Second-ballot management. When the execution status of a task $\mathcal{T}_1$ is uncommitted, a second-ballot can be enforced. The second-ballot is provided through a new task $\mathcal{T}_2$, that is connected to $\mathcal{T}_1$. The second-ballot task is generated with the aim to help consensus convergence. The second-ballot task is only enforced for tasks with typology $t = proposition$ and will be discussed in Section 3.4.

3.1 LiquidCrowd techniques for task management

The consensus on the result $\bar{A}$ of a task $\mathcal{T}$ is defined as an answer agreement of the members of the group $\mathcal{G}$ involved in the execution of $\mathcal{T}$. To this end, a *weighted-voting mechanism* is exploited to discriminate the contribution of each worker $\mathcal{W} \in \mathcal{G}$ in successfully completing the task $\mathcal{T}$. The answer of a worker $\mathcal{W}$ has a weight corresponding to her/his trustworthiness $\tau_{\mathcal{W}}$, so that the higher is the worker trustworthiness $\tau_{\mathcal{W}}$, the

higher is the weight of the answer provided by $\mathcal{W}$ in calculating the consensus. The idea of adopting a weighted-voting strategy for consensus evaluation is borrowed from the literature (see for instance [Demartini et al., 2012]) and it has the goal to limit the possible negative contribution of inaccurate workers in calculating the consensus. Furthermore, we rely on a *supermajority mechanism* to determine whether the answer that obtained the highest degree of consensus within the group $\mathcal{G}$ has *enough* weight (i.e., trustworthiness) for considering the assigned task $\mathcal{T}$ as successfully completed (i.e., committed).

In LiquidCrowd, consensus is *post-verified* at each execution $e_i \in E$, meaning that the set of answers $\mathcal{A}^*$ provided by the members of the group $\mathcal{G}$ in the execution e_i is completely collected before performing consensus verification. We call *1st-candidate-answer* CA^1 the candidate-answer with the highest degree of consensus which is selected to become the task result after the e_i execution (the candidate-answer equivalence relation depends on the task typology, see Section 3.3 for more details). We call $\mathcal{G}_{CA^1} \subseteq \mathcal{G}$ the *support group* of CA^1, namely the sub-group of $\mathcal{G}$ members providing an answer contained in CA^1 to the assigned task $\mathcal{T}$.

According to the supermajority mechanism of LiquidCrowd, V_{CA^1} becomes the task result $\bar{A}$ and the task is committed iff the following two constraints are satisfied.

Q-constraint. It is the *quorum constraint* checking that the consensus obtained by the 1st-candidate-answer CA^1 satisfies the quorum q. It ensures that a qualified majority selected an answer in CA^1 as task answer by checking that the trustworthiness of the support group $\mathcal{G}_{CA^1}$ is higher than the quorum percentage applied to the overall trustworthiness of all the members of the group $\mathcal{G}$, namely:

$$\sum_{\mathcal{W} \in \mathcal{G}_{CA^1}} \tau_{\mathcal{W}} \geq q \cdot \sum_{\mathcal{W} \in \mathcal{G}} \tau_{\mathcal{W}}$$

where $\sum_{\mathcal{W} \in \mathcal{G}_{CA^1}} \tau_{\mathcal{W}}$ is the trustworthiness of the support group $\mathcal{G}_{CA^1}$, and $\sum_{\mathcal{W} \in \mathcal{G}} \tau_{\mathcal{W}}$ is the trustworthiness of all the members of the group $\mathcal{G}$.

Bop-constraint. It is the *balance-of-power* constraint checking that a single worker cannot shift the majority from one answer to another one by changing her/his own task answer. The bop-constraint verifies that it is not possible to satisfy the q-constraint by

moving a worker in the support group of CA^1 to the support group of another candidate-answer, namely:

$$\sum_{\mathcal{W} \in \mathcal{G}_{CA^2}} \tau_{\mathcal{W}} + \tau_{\mathcal{W}}^{max} < q \cdot \sum_{\mathcal{W} \in \mathcal{G}} \tau_{\mathcal{W}}$$

where CA^2 is the *2nd-candidate-answer*, namely the candidate-answer with the highest degree of consensus just after CA^1, and $\tau_{\mathcal{W}}^{max}$ is the maximum trustworthiness value among the workers $\mathcal{W} \in \mathcal{G}_{CA^1}$.

The supermajority checking procedure is formalized in Algorithm 1.

Re-execution of uncommitted tasks. When the consensus is not reached in the execution e_i, the possible re-execution of the task $\mathcal{T}$ needs to be considered. When $i = k$, the maximum number of re-executions is reached, then the execution status s_{e_k} is set to terminated and the task is considered as unsuccessfully completed (the task re-execution is not possible). When $i < k$, the execution status s_{e_i} is set to uncommitted and a new execution e_{i+1} of the task $\mathcal{T}$ is scheduled. The new group of workers $\mathcal{G}^{i+1}$ is composed by selecting workers $\mathcal{W}$ where i) $\mathcal{W}$ did not participate to a previous execution e_j ($j \in [1,i]$) of the task $\mathcal{T}$, and ii) $\mathcal{W}$ has an appropriate trustworthiness value (i.e., $\tau_{\mathcal{W}} \geq th_{\tau}^{i+1}$). The trustworthiness threshold th_{τ} is progressively increased each time the task $\mathcal{T}$ is re-executed. At the first execution of $\mathcal{T}$, the trustworthiness threshold is $th_{\tau}^1 = 0$. A smooth increase of the trustworthiness threshold is recommended from one re-execution to the subsequent one (e.g., $th_{\tau}^{i+1} = th_{\tau}^i + 10\%$). This to avoid that top-trustworthy workers are massively involved in the execution of many tasks while mid-trustworthy workers are constantly excluded.

The introduced technique for task re-execution performs a selection of the workers with the required skills (i.e., capability to produce shareable answers): this goes in the direction proposed in [Roy et al., 2013] with the paradigm "who will be asked to contribute to what".

Example. Consider a group $\mathcal{G}^1$ with size $s_{\mathcal{G}^1} = 5$ and a task $\mathcal{T}_1$ with possible answers $PA = \{E, HS, PS, D\}$ (details about the task example will be provided in Section 6.4.1). At the first execution e_1, the set of answers $\mathcal{A}^*$ collected from the $\mathcal{G}^1$ members are the following:

Data: Current execution e_i, candidate-answers CA^1,CA^2, max number of re-executions k
Result: Execution status s_e
; // Check q-constraint
tot_trustworthiness $\leftarrow 0$;
foreach $\mathcal{W} \in \mathcal{G}$ **do**
  tot_trustworthiness $\leftarrow$ tot_trustworthiness + $\tau_{\mathcal{W}}$;
end
trustworthiness_th $\leftarrow$ tot_trustworthiness$\cdot q_{e_i}$;
max_trustworthiness_CA^1 $\leftarrow 0$;
foreach $\mathcal{W} \in \mathcal{G}_{CA^1}$ **do**
  trustworthiness_CA^1 $\leftarrow$ trustworthiness_CA^1 + $\tau_{\mathcal{W}}$;
  if $\tau_{\mathcal{W}} >$ *max_trustworthiness_*CA^1 **then**
    max_trustworthiness_CA^1 $\leftarrow \tau_{\mathcal{W}}$
  end
end
if *trustworthiness_*$CA^1 <$ *trustworthiness_th* **then**
  if $i < k$ **then**
    return uncommitted
  else
    return terminated
  end
end
; // Check bop-constraint
foreach $\mathcal{W} \in \mathcal{G}_{CA^2}$ **do**
  trustworthiness_CA^2 $\leftarrow$ trustworthiness_CA^2 + $\tau_{\mathcal{W}}$;
end
if *trustworthiness_*CA^2 + *max_trustworthiness_*$CA^1 \geq$ *trustworthiness_th* **then**
  if $i < k$ **then**
    return uncommitted
  else
    return terminated
  end
else
  return committed;
end

Algorithm 1: The supermajority algorithm for consensus verification in LiquidCrowd

Group $\mathcal{G}^1$ (execution e_1)	
$A^{\mathcal{W}_1}(\tau_{\mathcal{W}_1} = 0.7)$:	HS
$A^{\mathcal{W}_2}(\tau_{\mathcal{W}_2} = 0.3)$:	D
$A^{\mathcal{W}_3}(\tau_{\mathcal{W}_3} = 0.3)$:	HS
$A^{\mathcal{W}_4}(\tau_{\mathcal{W}_4} = 0.8)$:	HS
$A^{\mathcal{W}_5}(\tau_{\mathcal{W}_5} = 0.4)$:	PS

The 1st-candidate-answer is CA^1, where the representative value is $V_{CA^1} = [\mathsf{HS}]$, and the corresponding support group is $\mathcal{G}^1_{CA^1} = \{\mathcal{W}_1, \mathcal{W}_3, \mathcal{W}_4\}$. Considering a quorum $q = 0.51$, the q-constraint is satisfied by the candidate-answer CA^1, in that:

$$\left[\sum_{\mathcal{W} \in \mathcal{G}^1_{CA^1}} \tau_{\mathcal{W}} = 1.8\right] \geq \left[q \cdot \sum_{\mathcal{W} \in \mathcal{G}^1} \tau_{\mathcal{W}} = 1.275\right]$$

The 2nd-candidate-answer is CA^2, where $V_{CA^2} = [\mathsf{PS}]$ and $\mathcal{G}^1_{CA^2} = \{\mathcal{W}_5\}$. Moreover, we have $\tau^{max}_{\mathcal{W}} = \tau_{\mathcal{W}_4} = 0.8$. As a result, the bop-constraint is satisfied by the candidate-answer CA^1, in that:

$$\left[\sum_{\mathcal{W} \in \mathcal{G}^1_{CA^2}} \tau_{\mathcal{W}} + \tau_{\mathcal{W}_4} = 1.2\right] < \left[q \cdot \sum_{\mathcal{W} \in \mathcal{G}^1} \tau_{\mathcal{W}} = 1.275\right]$$

The execution status s_{e_1} is set to committed and the result of the task $\mathcal{T}_1$ is $\bar{A} = \mathsf{HS}$.

3.2 LiquidCrowd techniques for worker management

After completing a task execution, each worker $\mathcal{W} \in \mathcal{G}$ receives a salary $s \in [0,1]$ and the corresponding score $\sigma_{\mathcal{W}}$ is increased (i.e., $\sigma_{\mathcal{W}} = \sigma_{\mathcal{W}} + s$). We recall that workers always receive the salary after task execution, regardless the result of consensus verification. For committed tasks, reward assignment and trustworthiness update are also executed.

Award assignment. The award $a \in [0,1]$ is defined to reward the positive contribution of a worker in reaching the consensus and in committing the task $\mathcal{T}$. For this reason, only workers that provided a contribution identical/equivalent to $\bar{A}$ receive the award

as a bonus on their score, namely $\sigma_{\mathcal{W}} = \sigma_{\mathcal{W}} + a$ where $\mathcal{W} \in \mathcal{G}_{CA^1}$. The workers that provided a different answer from $\bar{A}$ do not receive the award and their score remains unchanged.

Trustworthiness update. The trustworthiness $\tau_{\mathcal{W}} \in [0,1]$ of a worker $\mathcal{W}$ is defined to capture the worker ability to foster the task commitment and it is based on the worker history in executing tasks. At the beginning of the crowdsourcing activities (time $t = 0$), the worker trustworthiness $\tau_{\mathcal{W}}$ is set to an initial value $\tau^0_{\mathcal{W}} = \tau_0$. Each time a task $\mathcal{T}$ is committed (time $t+1$), the trustworthiness of a worker $\mathcal{W} \in \mathcal{G}$ is updated as follows:

$$\tau^{t+1}_{\mathcal{W}} = \begin{cases} w_h \cdot \tau^t_{\mathcal{W}} + (1 - w_h) \cdot a, & \text{if } \mathcal{W} \in \mathcal{G}_{CA^1} \\ w_h \cdot \tau^t_{\mathcal{W}}, & \text{if } \mathcal{W} \notin \mathcal{G}_{CA^1} \end{cases}$$

where $\tau^t_{\mathcal{W}}$ is the trustworthiness value of the worker $\mathcal{W}$ before the execution of the task $\mathcal{T}$ (time t), and $w_h \in (0,1]$ is the *history weight*. For a worker $\mathcal{W}$, the calculation of $\tau^{t+1}_{\mathcal{W}}$ takes into account both the worker trustworthiness gathered in the history of all the task executed by $\mathcal{W}$ (i.e., $w_h \cdot \tau^t_{\mathcal{W}}$) and the (possible) award received in the execution of the last task $\mathcal{T}$. The worker trustworthiness increases (i.e., $\tau^{t+1}_{\mathcal{W}} > \tau^t_{\mathcal{W}}$) when $\mathcal{W} \in \mathcal{G}_{CA^1}$, meaning that the worker confirmed her/his ability to foster task commitment in the last-executed task $\mathcal{T}$, and it decreases otherwise. The history weight w_h allows to apply different strategies in determining the degree of trustworthiness update. Low values of history weight (e.g., $0.1 \leq w_h \leq 0.5$) are employed to enforce a *short-memory* strategy of trustworthiness update, where the new trustworthiness value $\tau^{t+1}_{\mathcal{W}}$ is mainly determined by the award a received for the execution of the last task $\mathcal{T}$. On the opposite, high values of trustworthiness update (e.g., $0.6 \leq w_h \leq 0.9$) are employed to enforce a *long-memory* strategy of trustworthiness update, where the new trustworthiness value $\tau^{t+1}_{\mathcal{W}}$ is mainly determined by $\tau^t_{\mathcal{W}}$, that is the worker trustworthiness gathered by $\mathcal{W}$ in the whole history of executed tasks before $\mathcal{T}$.

Example. Consider the example of Section 3.1. After the execution e_1 of the task $\mathcal{T}_1$, score and trustworthiness of $\mathcal{G}^1$ members are updated. Consider a salary $s = 0.1$ and an award $a = 1.0$. Each worker $\mathcal{W} \in \mathcal{G}^1$ receives the salary and $\sigma_{\mathcal{W}} = \sigma_{\mathcal{W}} + 0.1$. In addition, the workers in $\mathcal{G}^1_{CA^1} = \{\mathcal{W}_1, \mathcal{W}_3, \mathcal{W}_4\}$ receive the award. For instance, the score of the worker $\mathcal{W}_1$ is $\sigma_{\mathcal{W}_1} = \sigma_{\mathcal{W}_1} + 1$. Finally, the trustworthiness is updated for

all the workers in $\mathcal{G}^1$. Consider an history weight $w_h = 0.8$. The trustworthiness value of the worker $\mathcal{W}_1 \in \mathcal{G}^1_{CA^1}$ is $\tau^{t+1}_{\mathcal{W}_1} = w_h \cdot \tau^t_{\mathcal{W}_1} + (1 - w_h) \cdot a = 0.8 \cdot 0.7 + 0.2 \cdot 1 = 0.76$ ($\tau^{t+1}_{\mathcal{W}_1} > \tau^t_{\mathcal{W}_1}$). The trustworthiness value of the worker $\mathcal{W}_2 \notin \mathcal{G}^1_{CA^1}$ is $\tau^{t+1}_{\mathcal{W}_2} = w_h \cdot \tau^t_{\mathcal{W}_2} = 0.8 \cdot 0.3 = 0.24$ ($\tau^{t+1}_{\mathcal{W}_2} < \tau^t_{\mathcal{W}_2}$).

Rewarding considerations about the awarding mechanism of LiquidCrowd. A reference crowdsourcing definition recommends that a worker receives a reward according to the number of executed tasks for avoiding frustration and quality reduction in subsequent activities [Scekic et al., 2013]. To this end, we decided to introduce the salary component of the score σ, which is paid regardless of the task commitment. On the opposite, the award component of σ is introduced to encourage accurate task execution and it is assigned to the workers that contributed to reach the consensus. A possible criticism to such a kind of awarding mechanism is expressed by the following question: "*is it fair to reward a worker just for the fact that she/he participated to the consensus, namely she/he provided the most-voted task answer?*, *Why do you consider the most-voted answer as the best task result?*"

3.3 LiquidCrowd techniques for candidate-answers identification

By identification of candidate-answers, we mean the process of aggregating and combining worker answers into equivalence classes, based on the level of similarity/identity of the answers. The identification of an equivalence class $CA = \langle V, \mathcal{G}_{CA} \rangle$, is based on an equivalence relation $\mathcal{R}$. Different equivalence relations are introduced based on the degree of freedom that workers have in answering a task. For example, detecting the 1st-candidate-answer equivalence class is different if the task typology is $t = choice$ or $t = range$. To better understand this concept, consider for example, a task where the workers are asked to "*tell the elevation of the mountain K2, expressed in meters*". If the considered task has typology $t = choice$, the workers will be asked to choose among a set of predefined answers (e.g., *6452m*, *8114m*, *8611m*). In such a situation, the identification of candidate-answers is straightforward, since workers can only provide answers corresponding to one of the three predefined options. Suppose for example that the following answers have been gathered from a group of 6 workers:

8114m, *8114m*, *6452m*, *8611m*, *8611m*, *8611m*. By relying on the equivalence relation $\mathcal{R}$ =*answers are identical*, we have that *8611m* represents the equivalence class composed by 3 identical worker answers, which becomes the 1st-candidate-answer.

On the opposite, consider the same question proposed to the workers through a task with typology $t = range$. In this case, the workers have to produce an answer from scratch on the basis of their knowledge and their expertise. Suppose for example that the following answers have been gathered from a group composed by 6 workers: *5164m*, *5165m*, *8600m*, *8630m*, *8610m*, *8605m*. To identify the 1st-candidate-answer, it is clear that the equivalence relation used before for the choice task is not adequate. A human supervisor, can recognize that the last 4 contributions are similar (i.e., close each other with respect to the considered task) and they are part of the same equivalence class which originates the 1st-candidate-answer. Appropriate techniques are required to simulate the behaviour of the human supervision in recognizing the 1st-candidate-answer.

In the following sections, we introduce the proposed equivalence relations and techniques for candidate-answers identification for each task typology.

3.3.1 Choice task

Equivalence relation $\mathcal{R}_{choice}$ = "*Worker answers are identical*".

The identification of candidate-answers for choice tasks relies on the equality metric since a set of predefined values is only possible (see Algorithm 2). In the first step, we create a list L containing all the answers of the group members. In the second step, the first element of L (called $L[0]$) is extracted from the list. A candidate-answer CA_1 is created with a representative value $V_{CA_1} = L[0]$. Moreover, the worker who produced $L[0]$ is added to the support group $\mathcal{G}_{CA_1}$ of CA_1. In the third step, an iteration over the remaining items of L is performed in order to find answers equal to $L[0]$. When an answer equal to $L[0]$ is found, it is removed from the list L and the associated worker is added to $\mathcal{G}_{CA}$. In the fourth step, the answer $L[0]$ is removed from the list. If L is empty, the procedure is completed; otherwise, the procedure starts again from the second step.

After completion, the obtained candidate-answers are ordered by their associated degree of consensus, to identify the 1st-candidate-answer and the 2nd-candidate-answer.

Data: Current execution e_i,
Result: Candidate-answers CA
; // Answers list generation
L $\leftarrow$ [];
foreach $A^{\mathcal{W}_x} \in \mathcal{A}^*$ **do**
  put($L, A^{\mathcal{W}_x}$);
end
; // Candidate-answers identification
$i \leftarrow 1$;
while *L is not empty* **do**
  $A^{\mathcal{W}_x} \leftarrow L[0]$;
  $V_{CA_i} \leftarrow A^{\mathcal{W}_x}$;
  add($\mathcal{G}_{CA_i}, \mathcal{W}_x$);
  for $k \leftarrow 1$ **to** *size(L)-1* **do**
    if $L[k] == L[0]$ **then**
      $A^{\mathcal{W}_x} \leftarrow L[k]$;
      add($\mathcal{G}_{CA_i}, \mathcal{W}_x$);
      remove($L, L[k]$);
    end
  end
  add(CA, CA_i);
  $i \leftarrow i + 1$;
end
return CA;

Algorithm 2: The candidate-answers identification algorithm for tasks with $t =$ *choice*

3.3.2 Range task

Equivalence relation $\mathcal{R}_{range}$ = "*Worker answers have a standard deviation within a prefixed threshold* th_{stddev}".

When the executed task $\mathcal{T}$ has typology $t = range$, workers can provide a free-numeric answer. In this case, a conventional mechanism to identify the candidate-

answers is to calculate the arithmetic mean over all the collected worker answers [Surowiecki, 2005; Malone et al., 2010]. Such a solution is not adequate to capture/detect possible agreements over the worker answers and to distinguish different support groups for candidate-answer identification. We stress that in the range task, the answers provided by workers can be very skewed, thus possible outliers need to be considered. For this reason, we propose to employ the standard deviation as a mechanism to identify the candidate answers. The idea is to first calculate an initial average value of all the provided worker answers using the arithmetic mean and then to exploit the standard deviation to group workers answers "near enough" in the same candidate answer. In particular, we define a standard deviation threshold th_{stddev} to set the maximum distance between two worker answers for being included in the same candidate answer. The proposed technique is shown in the following and it is formalized in Algorithm 3.

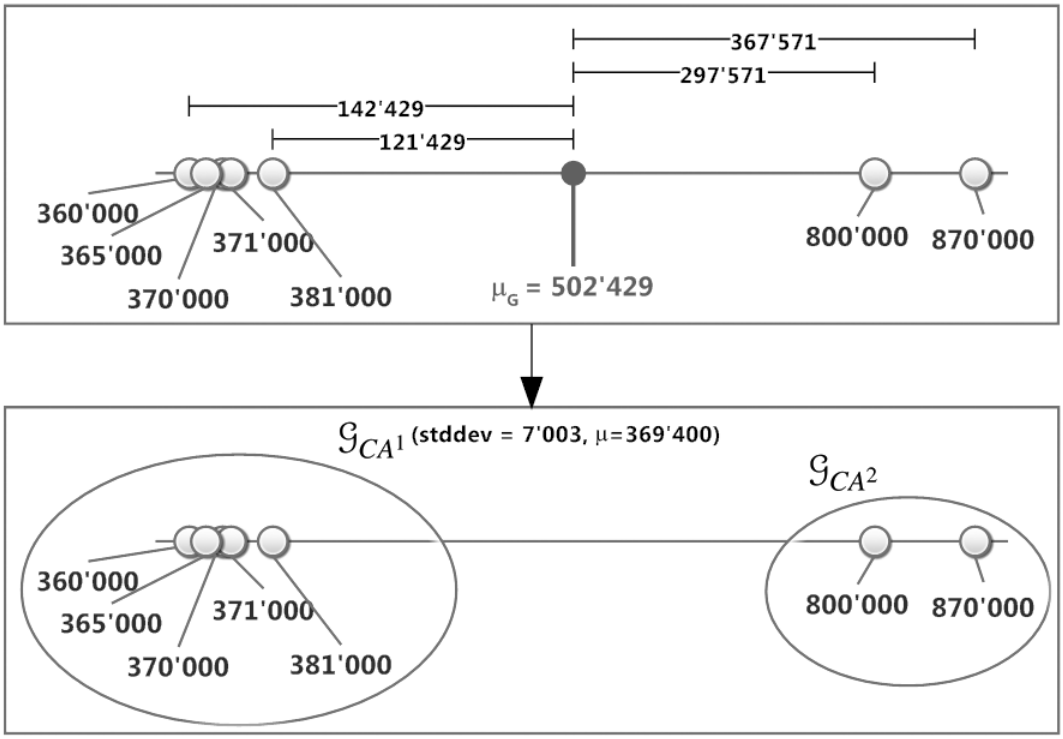


Figure 3.2: Candidate-answers identification in tasks with $t = range$

Identification of support groups. We present the following technique where two

support groups $\mathcal{G}_{CA^1}$ and $\mathcal{G}_{CA^2}$ are provided (see Figure 3.2).

1. Compute the arithmetic mean $\mu_{\mathcal{G}}$ over all the answers of the workers of group $\mathcal{G}$.
2. Compose the support group $\mathcal{G}_{CA^1}$ by selecting the answer being the closest to $\mu_{\mathcal{G}}$ with respect to the absolute value of the arithmetic subtraction.
3. Add to the support group $\mathcal{G}_{CA^1}$ the next answers being the closest to $\mu_{\mathcal{G}}$ and compute the arithmetic mean $\mu_{\mathcal{G}_{CA^1}}$.
4. Compute the the standard deviation for the answers in $\mathcal{G}_{CA^1}$ as

$$stddev = \sqrt{\frac{1}{|\mathcal{G}_{CA^1}|} \sum_{i=1}^{|\mathcal{G}_{CA^1}|} \left(A_i - \mu_{\mathcal{G}_{CA^1}}\right)^2}$$

   where $|\mathcal{G}_{CA^1}|$ represents the number of answers in $\mathcal{G}_{CA^1}$, and A_i, for $i \in \{1, \ldots, |\mathcal{G}_{CA^1}|\}$, represents the i^{th} answer in $\mathcal{G}_{CA^1}$.
5. If $stddev < th_{stddev}$ then go back to step 3. Otherwise, remove the last added contribution from $\mathcal{G}_{CA^1}$ and continue with step 6.
6. Compose the second support group $\mathcal{G}_{CA^2}$ by selecting all the remaining answers (i.e., all the answers of $\mathcal{G}$ that have not been added to $\mathcal{G}_{CA^1}$.

Evaluation of the representative value. Once the support group $\mathcal{G}_{CA^1}$ has been composed, we evaluate the 1st-candidate-answer representative value V_{CA^1} as the arithmetic mean over the answers in $\mathcal{G}_{CA^1}$, namely $V_{CA^1} = \mu_{\mathcal{G}_{CA^1}}$.

As a final remark, we note that we do not provide the 2nd-candidate-answer CA^2 because it is not necessary to comply with the supermajority mechanism. Indeed, in order to perform q-constraint and bop-constraint, we only need the 1st-candidate-answer representative V_{CA^1}, the 1st-candidate-answer support group $\mathcal{G}_{CA^1}$, and the support group of the 2nd-candidate-answer $\mathcal{G}_{CA^2}$.

Example. Consider a task $\mathcal{T}_1$ where workers are asked to guess the distance between earth and moon in kilometres. In such a context, the expected difference between

Data: Current execution e_i, standard deviation threshold th_{cv}
Result: Candidate-answers CA^1
; // Computation of the arithmetic mean $\mu_{\mathcal{G}}$
$\mu_{\mathcal{G}} \leftarrow avg(\mathcal{A}^*)$;
; // Identification of $\mathcal{G}_{CA^1}$
$\mathcal{G}_{CA^1} \leftarrow \emptyset$;
$stddev \leftarrow 0$;
$answers_{CA^1} \leftarrow$ empty array;
$A^{\mathcal{W}_x} \leftarrow$ getClosestAnswer($\mathcal{A}^*, \mu_{\mathcal{G}}$);
$\mathcal{A}^* \leftarrow \mathcal{A}^* / \{A^{\mathcal{W}_x}\}$;
add($\mathcal{G}_{CA^1}, A^{\mathcal{W}_x}$);
while $stddev < th_{stddev}$ ***and*** $\mathcal{A}^* \neq \emptyset$ **do**
  $A^{\mathcal{W}_x} \leftarrow$ getClosestAnswer($\mathcal{A}^*, \mu_{\mathcal{G}}$);
  $\mathcal{A}^* \leftarrow \mathcal{A}^* / \{A^{\mathcal{W}_x}\}$;
  add($answers_{CA^1}, A^{\mathcal{W}_x}$);
  $\mu_{\mathcal{G}_{CA^1}} \leftarrow avg(\mathcal{G}_{CA^1})$;
  $n \leftarrow |answers_{CA^1}|$;
  $stddev \leftarrow \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(answers_{CA^1}[i] - \mu_{\mathcal{G}_{CA^1}}\right)^2}$;
  if $stddev < th_{stddev}$ **then**
    $\mathcal{G}_{CA_1} \leftarrow \mathcal{G}_{CA_1} \cup \{\mathcal{W}_x\}$;
  else
    $V_{CA^1} \leftarrow \mu_{\mathcal{G}_{CA^1}}$;
    return CA^1;
  end
end
return CA^1;

Algorithm 3: The 1-st-candidate-answer identification algorithm for tasks with $t = range$

the worker contributions can be very high, due to the order of magnitude of the requested value (i.e., 369'453 Km). Consider the following set of gathered contributions: $\mathcal{A}^* = \{360'000, 365'000, 370'000, 371'000, 381'000, 800'000, 870'000\}$. Fol-

lowing the procedure presented in Section 3.3.2, we compute the arithmetic mean $\mu_{\mathcal{G}}$ as follows

$$M_{\mathcal{G}} = \frac{360000+365000+370000+371000+381000+800000+870000}{7} = 502429$$

In the next step, the support group $\mathcal{G}_{CA^1}$ is composed by iteratively selecting the contribution being the nearest to $\mu_{\mathcal{G}}$, checking the standard deviation $stddev$. Considering a standard deviation threshold $th_{stddev} = 10'000$, we obtain the support groups presented in Figure 3.2, where the standard deviation of the support group of CA^1 is $stddev(\mathcal{G}_{CA^1}) = 7'003$ and the arithmetic mean $\mu_{\mathcal{G}_{CA^1}} = V_{CA^1} = 369'400$.

Another issue relevant for the proposed technique is the choice of an appropriate threshold th_{stddev}. Consider the following example in comparison with the previous one.

Example. Consider a task $\mathcal{T}$ executed by a group $\mathcal{G}$ where workers are asked to guess the distance between the cities of Rome and Milan in kilometres (i.e., 476 Km). In such a context, the expected difference between the worker contributions is definitely lower than the example of Section 3.3.2 due to the lower order of magnitude of the expected answer. Consider the following set of gathered contributions: $\mathcal{A}^* = \{400, 450, 500, 512, 4000, 8000\}$. If we set a standard deviation threshold th_{stddev} equal to the one used in Section 3.3.2, we will obtain that all the provided contributions become part of the group $\mathcal{G}_{CA^1}$, due to the fact that their standard deviation is $stddev(\mathcal{A}^*) = 2853$.

We note that the standard deviation threshold should be manually set on a per-task basis, considering the maximum accepted average difference between the worker contributions. This have to be done considering two main aspects: i) the required worker precision and ii) the order of magnitude of the expected answer.

3.3.3 Proposition task

Equivalence relation $\mathcal{R}_{proposition}$ = "*Worker answers are syntactically similar*".

When the executed task $\mathcal{T}$ has typology $t = proposition$, workers provide free-text answers without any a-priori suggestion on how to answer a task. This means that

gathered answers can be very different in their syntax, even if they represent the same real-word concept (e.g., *science-fiction novel* and *sci-fi romance*). To face these kind of problems, text-analysis techniques are required to reconciliate and compare the worker answers. We classify available techniques in three classes:

- **Soft-normalization techniques**. Such techniques do not change the syntactic structure of the answers. Examples of these techniques are trimming and lower-casing. These techniques are always applied in the LiquidCrowd approach, because they do not produce any kind of ambiguity.
- **Hard-normalization techniques**. Techniques in this class modify the syntactic structure of the answers. Examples of these techniques are stop-words removal, stemming and lemmatisation. We do not apply these techniques in LiquidCrowd because they can lead to ambiguous situations. Consider, for example, the following case where a stop-word removal technique is applied to the answers provided by a group $\mathcal{G}$ of 4 workers asked to write the name of the famous pop-music group lead by Gary Barlow. The workers provide the following answers: take that, take one, take one, queen. If we apply a stop word removal technique in this context, the first 3 answers become take (i.e., due to the removal of the stop-word that). This is obviously incorrect and it is due to the ambiguity introduced by the stop-word removal technique.
- **Similarity-matching techniques**. Techniques in this class rely on a similarity metric to perform a syntactic comparison among each pair of answers. Examples of available metrics are n-gram, Levenshtein, cosine-similarity [Navarro, 2001]. We do not apply these techniques in LiquidCrowd because they can generate ambiguities. Examples of unsuccessful application of these techniques are those words having similar structure but different meaning (e.g., Carbon and Carton). By applying similarity techniques to this kind of words, we obtain high similarity values generating ambiguities.

After the application of soft-normalization techniques, the procedure for the identification of the candidate-answers is the same of the one shown for choice tasks (see Section 3.3.1), formalized in Algorithm 2.

3.4 Considerations about task typologies

In this section, we discuss some critical aspects related to task execution for each task typology supported in LiquidCrowd.

Choice. Managing answers in the choice task is straightforward due to the limited degree of freedom of involved workers. Possible problems in aggregating consensus can be due to possible ambiguities in the set of possible answers. Depending on the test case, it can happen that two or more options are very similar or misleading, thus causing worker confusion. The selection of the set of possible answer *PA* is a crucial step to avoid high numbers of uncommitted tasks. The use of the second-ballot is very uncommon in task with $t = choice$, and it can be enforced when the set of possible answers is composed by a large number of options (e.g., $|PA| > 5$), to ensure a focused consensus on the committed answer.

Range. In Section 3.3.2, we proposed a technique based on the arithmetic mean to identify the candidate-answers for range tasks. Other techniques can be adopted to identify candidate answers, such as unsupervised clustering [Xu and Wunsch II, 2005] and outliers detection [Hodge and Austin, 2004]. Even if such techniques are suitable for reflecting the agreement of the workers, we decided to adopt a technique based on the standard deviation and on the notion of support group. The main motivation is related to the fact that the number of answers provided by a group of workers is generally low (e.g., few units/a dozen) and techniques based on clustering/outlier-detection are generally more recommended for large sets of individuals and would introduce a useless computational overhead in our case. Furthermore, to validate the choice of standard deviation, we performed experimentations of the technique on "toy test-cases" built on tasks whose answer was known a-priori (e.g., "Which is the height of mountain K2" or similar). We simulated a set of random answers for each task and we applied the proposed technique based on standard deviation with support groups to identify the corresponding candidate answers and task results. In all the cases we found that the final task results produced by our technique were very close to the expected answer. The example shown in Figure 3.2 is taken from the test case related to K2 height. Based on all these considerations, current prototype of LiquidCrowd developed for the PhD Thesis implements the standard deviation technique. However, future versions of

LiquidCrowd could be based on clustering/outlier-detection techniques as discussed in future work Chapter.

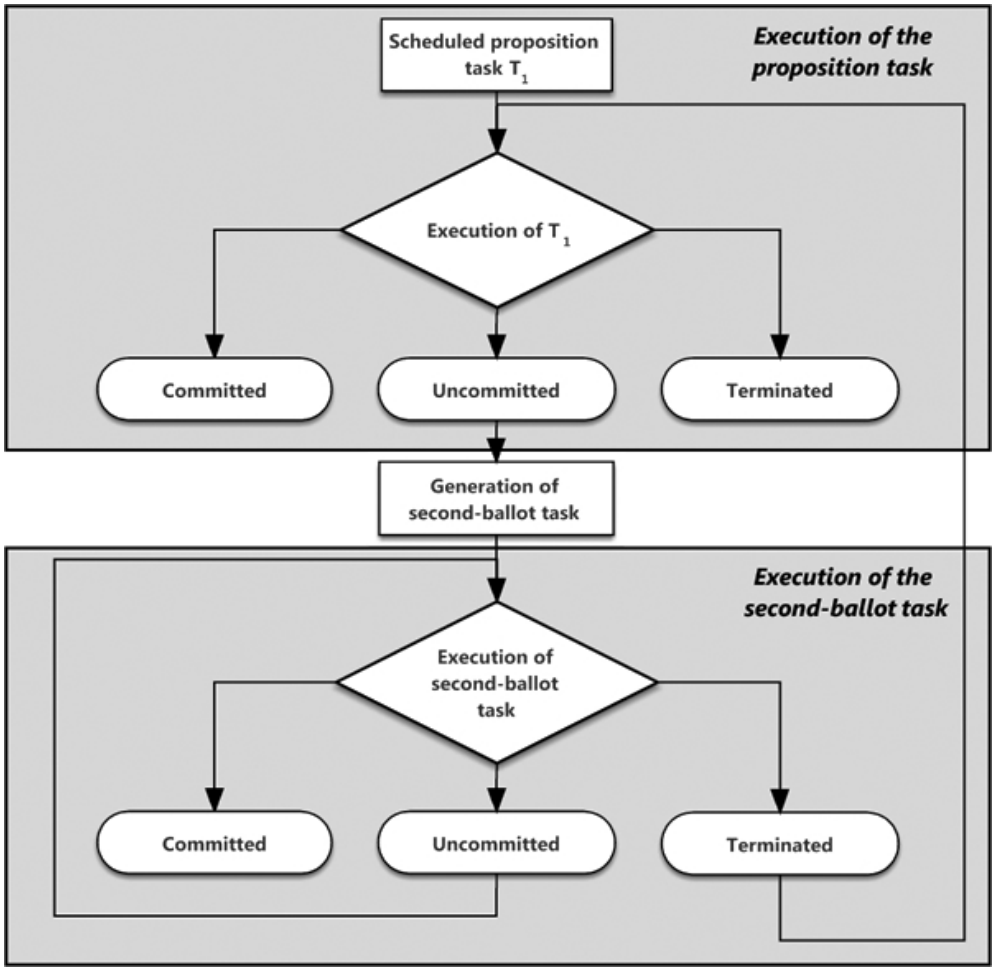


Figure 3.3: The second-ballot management schema

Proposition. This is the task typology that enforces the highest degree of freedom to the workers involved in the task execution. The main problem related to proposition tasks is the capability to effectively identify similar answers forming an equivalence class. In this kind of tasks, worker contributions can be syntactically different even if they represent the same real world entity/concept, making consensus evaluation very difficult. To overcome this problem, we introduce a second-ballot task with the aim of

reducing the degree of freedom of the proposition task. This is performed through the generation of a child task $\mathcal{T}_2$ with typology $t = choice$ where the most voted contributions of the parent task become the possible answers of $\mathcal{T}_2$. The second-ballot management procedure is presented in Figure 3.3 and it is based on the following steps. When a task $\mathcal{T}_1 = \langle \emptyset, S_1, E_1, \bar{A} \rangle$ with a structure $S_1 = \langle d_1, c_1, \text{“proposition”}, PA_1 \rangle$ is in state uncommitted, a second-ballot task $\mathcal{T}_2 = \langle \mathcal{T}_1, S_2, E_2, \bar{A} \rangle$ with $S_2 = \langle d_2, c_1, \text{“choice”}, PA_2 \rangle$ is generated. The newly generated task $\mathcal{T}_2$ has the same context c_1 of $\mathcal{T}_1$. The description of the second-ballot task d_2 is automatically generated by appending an explanatory text at the beginning of the description d_1 (i.e., “Second ballot: choose the best option among the answers produced for the question $[d_1]$”). Moreover, the parent task of $\mathcal{T}_2$ is set to $\mathcal{T}_1$. Finally, the associated set of possible answers $PA_2 \subseteq PA_1$, is composed by the most voted answers of $\mathcal{T}_1$. The second-ballot task $\mathcal{T}_2$ follows the executions steps presented in Figure 3.1, until the last execution status s_e is committed or terminated. When the execution status is committed, the final-result of the original task $\mathcal{T}_1$ is set to the final result of $\mathcal{T}_2$, and the last execution status s_e of $\mathcal{T}_1$ is changed to committed. Otherwise, the original proposition task is sent back for re-execution. The second ballot generation procedure continues until the task $\mathcal{T}_1$ ends with an execution status s_e equal to committed or a second-ballot task ends with terminated and the number of executions of $\mathcal{T}_1$ is equal to the maximum re-executions k.

Retroactive rewarding. When the result of an execution of a proposition task $\mathcal{T}_1$ is uncommitted, all the workers in the group $\mathcal{G}_{\mathcal{T}_1}$ only receive the salary (see Section 3.2). However, it is possible that the task $\mathcal{T}_2$ (child task of $\mathcal{T}_1$) is committed on an answer $\bar{A}$. In this case, the member of the group $\mathcal{G}_{\mathcal{T}_2}$ responsible of the $\mathcal{T}_2$ execution are rewarded (see Section 3.2). In addition to this, we also want to reward those workers of the group $\mathcal{G}_{\mathcal{T}_1}$ responsible of the $\mathcal{T}_1$ execution to award the fact they proposed the answer $\bar{A}$ finally selected to commit $\mathcal{T}_2$. To this end, we introduce a retroactive rewarding assigned to the workers who proposed a contribution identical/equivalent to the committed answer.

Example. Consider a task $\mathcal{T}_1$, executed by a group of 5 workers, each of them having a trustworthiness value $\tau_{\mathcal{W}} = 1.0$, asked to guess the best category fitting a set of books written by Isaac Asimov. The following contributions are gathered after the first execution e_1: “Science-fiction novel”, “sci-fi romance”, “Science-fiction novel”, “sci-fi”, “sci-fi romance”. We have that the q-constraint is not satisfied due to the

inability for automatic techniques to recognize the equivalence between the considered answers. We also note that it would be quite simple for human workers to choose "Science-fiction novel" as the most complete answers with respect to "sci-fi romance" or "sci-fi". The current task execution status s_{e_i} is set to uncommitted, thus a second-ballot task $\mathcal{T}$ is generated. The task $\mathcal{T}$ has a new description (i.e., asking to choose the best option among the answers produced to the question "Guess the best category fitting a set of books written by Isaac Asimov"), the same context of $\mathcal{T}_1$ (i.e., a void context), and a typology $t =$"choice". The set *PA* of possible answers is obtained by selecting the top-k most voted answers of $\mathcal{T}_1$. In this case, choosing the top-2 answers, the set of possible answers of $\mathcal{T}_2$ is $PA_2 = \{$"Science-fiction novel","sci-fi romance"$\}$. After the execution of $\mathcal{T}_2$, we have three possible scenarios.

1. $\mathcal{T}_2$ *is committed (e.g.,* $\bar{A} =$ *"Science-fiction novel").* In this case, no re-executions are needed and the last execution status of $\mathcal{T}_1$ is set to committed. Moreover, the workers that proposed a contribution that is identical/equivalent to $\bar{A}$ receive the award.

2. $\mathcal{T}_2$ *is* uncommitted. In this case, the task $\mathcal{T}_2$ is sent back for re-execution.

3. $\mathcal{T}_2$ *is* terminated. In this case, the task $\mathcal{T}_1$ is sent back for re-execution.

Considering the third scenario, the second-ballot procedure is executed until a second-ballot task having $\mathcal{T}_1$ as parent task ends with an execution status equal to terminated after $\mathcal{T}_1$ reached the maximum re-executions k.

Chapter 4

The Argo crowdsourcing prototype

The LiquidCrowd techniques have been implemented in the Argo prototype which will be used for the application of LiquidCrowd approach to data classification problems presented in Chapter 6.

4.1 The Argo motivations and features

The Argo prototype is the proposed solution for the application of LiquidCrowd techniques. With respect to the classification framework of Section 2.1, Argo falls in the field of independent solutions, since it is not based on any labour market.

The motivations behind the decision to implement an independent crowdsourcing platform can be summarized as follows.

- *Rewarding mechanism*. There are two main motivations related to worker rewarding for choosing an independent platform: 1) the need for a *retroactive awarding* for tasks with typology $t = proposition$, and 2) the decision to rely on *incentives* different from money.

  About 1), as discussed in Section 3.4, workers involved in an uncommitted proposition task receive the salary and they can receive a retroactive award after the commitment of a linked second-ballot task. This feature is not available in market-based crowdsourcing platforms: in fact, after the completion of a task,

the requester should decide if the received contribution is valuable or not and if the worker should be rewarded or not. No further payment actions are provided.

About 2), we decided to publish the names of the best 10 workers (sorted by their score) with our acknowledgements on the LiquidCrowd Facebook page. Moreover, we relied on the interest of the students in participating to a scientific experimentation. Referring to [Malone et al., 2010], these incentives are described as *glory* and *love*.

- *Target crowd*. For the application of LiquidCrowd, we decided to rely on two pre-defined groups of workers: a group composed by students of the Master Degree in Informatics, and a group composed by students of the Master Degree in Humanities. In such a context, the use of a market-based solution was not suitable due to the inability to restrict the target workers to a pre-defined group of people. In Argo, worker registration has been restricted to the people involved in the experimentations.

The Argo prototype is an independent crowdsourcing platform providing worker recruitment/retention, consensus-based task validation mechanisms, and worker trustworthiness assessment mechanisms. More in detail, Argo is characterized by the following features.

Group-based task scheduling. A task to be executed is queued for scheduling until a sufficient number of workers is available to compose a group with appropriate trustworthiness requirements. Given the group size $s_{\mathcal{G}}$ and a task $\mathcal{T}$ to be executed with trustworthiness threshold th_{τ}, $\mathcal{T}$ is scheduled when $s_{\mathcal{G}}$ workers in the status $s_{\mathcal{W}} = available$ are detected to compose a group $\mathcal{G}$ and each worker $\mathcal{W} \in \mathcal{G}$ has $\tau_{\mathcal{W}} \geq th_{\tau}$. A worker $\mathcal{W} \in \mathcal{G}$ is unavailable until she/he has executed the assigned task $\mathcal{T}$. Moreover, a worker can set her/his status to unavailable with the aim to (temporarily) stop her/his involvement in the crowdsourcing activities. When all the required workers are eventually available, the task is executed following the LiquidCrowd approach presented in Figure 3.1.

Supermajority-based consensus verification. Argo implements all the techniques for consensus verification presented in Section 3. Candidate answers are identified following the techniques presented in Section 3.3 for the three different task typologies

(i.e., choice, rate, proposition). Finally, q-constraint and bop-constraint are checked as explained in Section 3.1.

Salary-based and award-based worker rewarding. In Argo, each worker is associated with a score σ_W. The worker score can be incremented in two ways: i) by submitting a contribution (i.e., receiving the salary), and ii) by being part of the consensus for a task (i.e., receiving the award). The Argo prototype also enforces the retroactive rewarding for proposition tasks.

Competition-oriented score publication. In Argo, score and trustworthiness of top-k workers are visible to all the crowdsourcing workers as a sort of "hall-of-fame". This to encourage positive competition and participation to task execution. Score and trustworthiness values can be anonymized when published. A worker can choose this option in her/his personal profile for privacy reasons.

Data persistence. The Argo prototype is paired with a relational DBMS used to persist all the data related to task executions and worker attributes (e.g., worker contributions, final task results, worker trustworthiness and score). This way, all the information related to the crowdsourcing process can be reused for further experimentations and studies.

4.2 The Argo architecture

The architecture of the Argo prototype is presented in Figure 4.1 and it is divided in four main managers: i) the task manager, ii) the consensus manager, iii) the reward manager, and iv) the uncommitment manager. Each manager performs isolated operations and communicates with other managers through the LiquidCrowd database. In the following, the main four managers functionalities of Argo are presented in detail.

4.2.1 Task manager

In Argo, the task assignment is triggered by the workers: in fact, when a worker decides to execute a new task she/he can request a new task assignment that will be accomplished by the task manager. The task manager performs two main operations:

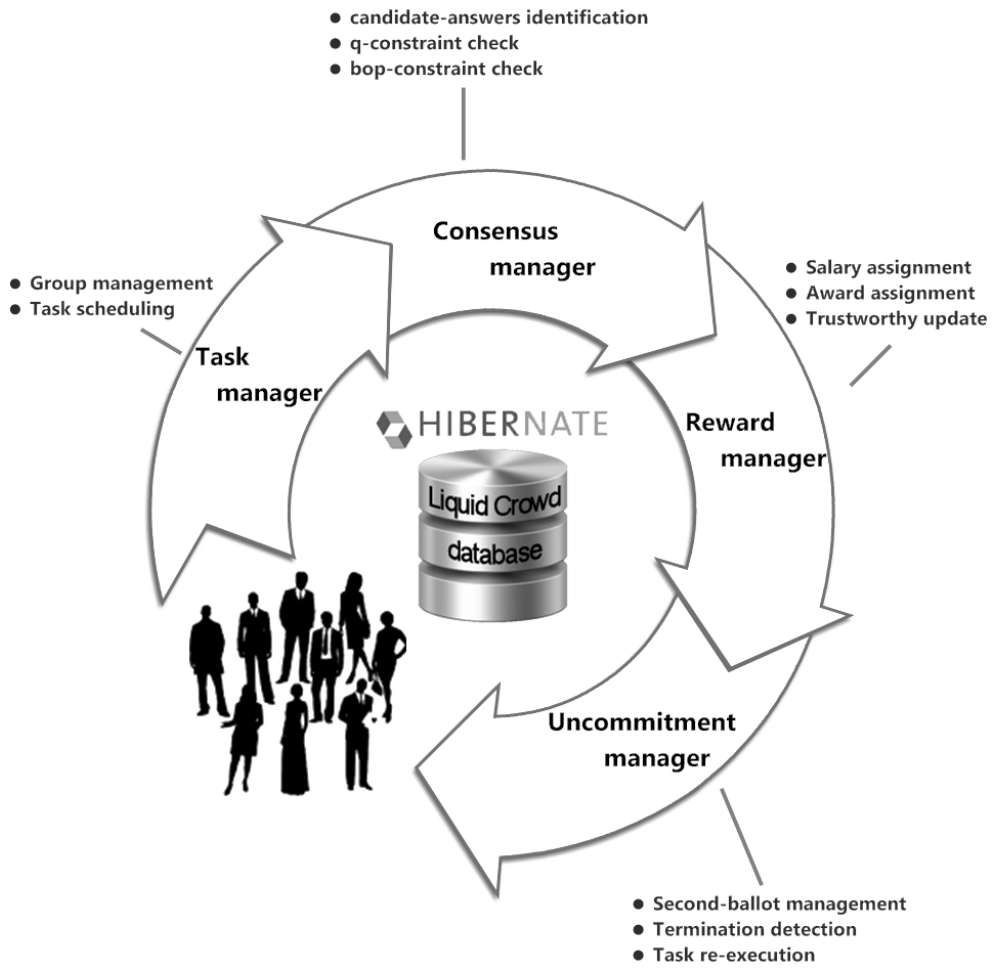


Figure 4.1: The Argo architecture

group management and *task scheduling*.

Group management. This includes all the necessary operations for creating groups, verifying the required number of workers in each group, triggering the consensus manager for completed groups and closing groups. In Argo, a task needs to be executed when it is in an uncommitted state, namely when: i) it is a new task (submitted to the system by the requester or generated as a second-ballot task), or ii) it is an uncommitted task that needs re-execution. For each task $\mathcal{T}$ that needs to be executed, the task manager creates an empty group $\mathcal{G}_{\mathcal{T}}$ assigned to the task. When a worker $\mathcal{W}$ requires

a new task to execute, the task manager chooses a suitable task $\mathcal{T}$ (see *Task scheduling* below) and adds the worker $\mathcal{W}$ to a group $\mathcal{G}_\mathcal{T}$. After being added to a group, the worker $\mathcal{W}$ is able to perform the assigned task and submit her/his contribution, independently from the actions performed by the other workers of the group $\mathcal{G}_\mathcal{T}$ (i.e., contribution submission is asynchronous for the workers of a group). Moreover, the worker $\mathcal{W}$ can *reject* the task: this way, the rejected task will never be proposed to the worker $\mathcal{W}$ again. After task execution, each submitted contribution is stored in the LiquidCrowd database until the required number of contributions $s_\mathcal{G}$ have been gathered. As a further remark, we point out that the worker $\mathcal{W}$ assigned to the group $\mathcal{G}_\mathcal{T}$ cannot be added to another group $\mathcal{G}_{\mathcal{T}'}$ until she/he submit her/his contribution for $\mathcal{G}_\mathcal{T}$. After contribution submission, the worker $\mathcal{W}$ can continue participating in the crowdsourcing activities by requesting another task and consequently being added to another group. In other words, a worker can execute only one task at a time.

Moreover, when the required number of workers $s_\mathcal{G}$ have been added to $\mathcal{G}_\mathcal{T}$ and all the contributions have been gathered, the task manager triggers a *consensus verification request* for the group $\mathcal{G}_\mathcal{T}$ to the consensus manager and it closes the group $\mathcal{G}_\mathcal{T}$. After group closing, no other workers can be added to the group and all the information related to the group (e.g., workers contributions, assigned task) in maintained in the LiquidCrowd database.

Task scheduling. The task manager is also responsible for deciding the order of tasks submission to the workers. This is done by associating a priority-value with each task $\mathcal{T}$, which is computed by considering the following two aspects: a) the more workers are already associated with the group $\mathcal{G}_\mathcal{T}$, the higher the task priority will be, and b) second-ballot tasks have higher priority with respect to other tasks. These aspects are considered to speed-up the crowdsourcing process and to avoid worker dispersion over a high number of tasks. In other words, workers should be concentrated on tasks that are near to the completion (e.g., where the number of gathered contributions is near to the requested group size $s_\mathcal{G}$) in order to achieve the crowdsourced results as soon as possible.

Given a worker $\mathcal{W}$ requiring a task for execution, a two-step procedure composed by *task selection* and *task ordering* is performed. The task selection phase retrieves all the tasks to be executed that comply with the following requirements: i) the task have a trustworthiness threshold $th_\tau < \tau_\mathcal{W}$, ii) the task group $\mathcal{G}_\mathcal{T}$ does not contain the worker

$\mathcal{W}$, iii) the task has not been already rejected by the worker $\mathcal{W}$, and iv) the worker was not in the group that answered the parent task (i.e., when the considered task is a second-ballot). About i), this is a necessary requirement to ensure that workers with a trustworthiness value $\tau_{\mathcal{W}}$ could only execute tasks with a trustworthiness threshold $th_{\tau} < \tau_{\mathcal{W}}$. About ii) and iii), these requirements are needed to avoid that a worker could provide multiple contributions for the same task and avoid an already reject task assignment. About iv), this is done to avoid the self-support of the workers that took part to a proposition task in the second-ballot (i.e., a worker would probably vote for his/her own answer in order to increase the probability of obtaining the reward). After the task selection phase, the retrieved tasks are ordered by considering a) and b) introduced above; finally the first-ordered task is assigned to the worker $\mathcal{W}$.

4.2.2 Consensus manager

The consensus manager execution is triggered for a group by the task manager when the required number of contributions $s_{\mathcal{G}}$ for a task $\mathcal{T}$ have been gathered in $\mathcal{G}_{\mathcal{T}}$. The consensus manager implements the supermajority mechanism through the following three operations: *candidate-answers identification*, *q-constraint check* and *bop-constraint check*.

Candidate-answers identification. The consensus manager retrieves from the LiquidCrowd database all the worker contributions provided for the group $\mathcal{G}_{\mathcal{T}}$ and it applies the techniques presented in Section 3.3. More in detail, the following procedure is executed: i) each contribution in $\mathcal{G}_{\mathcal{T}}$ is associated with the trustworthiness value of the proposing worker; ii) the contributions are grouped in candidate-answers relying on their equality/equivalence; iii) each candidate-answer is associated with a value representing the overall trustworthiness of the contained contributions; iv) the candidate-answers are sorted by the associated trustworthiness value. Finally, the 1st and the 2nd-candidate-answers CA^1 and CA^2, are submitted the q-constraint and the bop-constraint checks.

Q-constraint check. The 1st-candidate-answer CA^1 is checked in order to verify that its trustworthiness value satisfies the quorum q. This is implemented by relying on the first part of the procedure formalized in Algorithm 1. Also in this case, the miss-

ing data (i.e., the overall trustworthiness of the workers in $\mathcal{G}_\mathcal{T}$) is retrieved from the LiquidCrowd database. This procedure has two possible outcomes: on one hand, if the q-constraint is satisfied, the bop-constraint check operation is invoked on CA^1 and CA^2, respectively. On the other hand, if the q-constraint is not satisfied, the task execution is marked as uncommitted and the reward manager execution is triggered on $\mathcal{G}_\mathcal{T}$.

Bop-constraint check. This operation is performed to check that no one of the workers in the support group of the CA^1 could make the CA^2 satisfy the q-constraint just by changing his own contribution (see Section 3.1 for further details). This is implemented by relying on the procedure formalized in the second part of Algorithm 1. This procedure has two possible outcomes: i) committed or ii) uncommitted. In both cases, the execution of the reward manager is triggered for the group $\mathcal{G}_\mathcal{T}$ and the outcome of this procedure is stored on the LiquidCrowd database.

4.2.3 Reward manager

The reward manager execution is triggered for a group $\mathcal{G}_\mathcal{T}$ by the consensus manager when consensus verification has been performed. The reward manager performs three operations: *Salary assignment*, *Award assignment*, and *Trustworthiness update*. The salary assignment operation is performed independently from the result produced by the consensus manager, while the award assignment and the trustworthiness update operations are performed only for tasks whose last execution status is s_e =committed.

Salary assignment. The salary assignment is performed for each worker $\mathcal{W} \in \mathcal{G}_\mathcal{T}$, irrespective of the result produced by the consensus manager (see Section 3.2). After the execution of the salary assignment, the score $\sigma_\mathcal{W}$ of each worker $\mathcal{W}$ is incremented by the salary s, namely $\sigma_\mathcal{W} = \sigma_\mathcal{W} + s$. After the salary assignment, if the last execution status is e_i =uncommitted, the execution of the uncommitment manager is triggered for the group $\mathcal{G}_\mathcal{T}$. Otherwise, the award assignment and the trustworthiness update operations are performed.

Award assignment. If the current execution status is s_e =Committed, *direct rewarding* and *retroactive rewarding* are performed as follows.

Direct rewarding is performed for each worker $\mathcal{W} \in \mathcal{G}_\mathcal{T}$ that provided a contribution identical/equivalent to the committed answer $\bar{A}$ (see Section 3.2). More in detail, the score $\sigma_\mathcal{W}$ of each worker $\mathcal{W} \in \mathcal{G}_{CA^1}$ is incremented with the award a, namely $\sigma_\mathcal{W} = \sigma_\mathcal{W} + a$.

Retroactive rewarding is performed only if the typology of the task $\mathcal{T}$ is equal to $t_\mathcal{T} = choice$ and the typology of the parent task P is $t_P = proposition$. In this case, the award assignment procedure explained above, is also applied to each worker $\mathcal{W} \in \mathcal{G}_P$, where $\mathcal{G}_P$ is the group assigned to the last execution of the task P (see Section 3.4). More in detail, the score $\sigma_\mathcal{W}$ of each worker $\mathcal{W} \in \mathcal{G}_P$ having submitted a contribution identical/equivalent to $\bar{A}$, is incremented with the award a, namely $\sigma_\mathcal{W} = \sigma_\mathcal{W} + a$.

Trustworthiness update. If the execution $s_e =$Committed, the trustworthiness value $\tau_\mathcal{W}$ of each worker $\mathcal{W} \in \mathcal{G}_\mathcal{T}$ is updated relying on the technique presented in Section 3.2. More in detail, the update is performed as follows:

$$\tau_\mathcal{W}^{t+1} = \begin{cases} w_h \cdot \tau_\mathcal{W}^t + (1 - w_h) \cdot a, & \text{if } \mathcal{W} \in \mathcal{G}_{CA^1} \\ w_h \cdot \tau_\mathcal{W}^t, & \text{if } \mathcal{W} \notin \mathcal{G}_{CA^1} \end{cases}$$

Finally, the execution of the uncommitment manager is triggered for the group $\mathcal{G}_\mathcal{T}$.

4.2.4 Uncommitment manager

The uncommitment manager execution is triggered for a group $\mathcal{G}_\mathcal{T}$ by the reward manager after the reward assignment and trustworthiness update. The uncommitment manager performs three operations: *termination detection*, *task re-execution*, and *second-ballot management*. These operations are performed only for tasks where the last execution status is $e_i =$uncommitted.

Second-ballot management. If the current execution status is $e_i =$uncommitted and the typology of the task $\mathcal{T}$ is $t = proposition$, a second-ballot task is generated. The uncommitment manager creates a new task C with typology $t = choice$. The context c and the description d of the task C are copied from the task $\mathcal{T}$. Finally the possible answers PA_C of the task C are set by selecting the top-3 voted answers of the last

execution e_i of the task $\mathcal{T}$. More in detail, the following procedure is executed for PA_C generation

1. Sort the candidate answers obtained from $\mathcal{G}_{\mathcal{T}}$ by the associated trustworthiness value
2. Add to PA_C the top-3 ordered answers
3. If the last-added answer has a trustworthiness value identical to the next-sorted answer, than add the next-sorted answer to PA_C. Repeat this step until the last-added answer has a trustworthiness value different from the next-sorted answer or until no more answers are available.

Termination detection. In this step, the number of past executions of $\mathcal{T}$ is checked against the maximum number of task executions k. More in detail, if the number of executions i of the task $\mathcal{T}$ is equal to the maximum number of task executions k, the current execution status e_i is set to terminated. In this case, the last execution status e_i is updated on the LiquidCrowd database.

Task re-execution. If the current execution status is e_i =uncommitted after the termination detection step, the task $\mathcal{T}$ have to be re-executed. The task $\mathcal{T}$ is marked for re-execution and the LiquidCrowd database is updated consequently.

4.3 Implementation issues

The Argo prototype has been developed as a web application in Java. It is based on two main technologies: the Spring Framework and the Hibernate ORM [Arthur and Azadegan, 2005; Bauer and King, 2006]. The Spring Framework provides a comprehensive programming and configuration model for Java-based applications. The Hibernate ORM is an Object/Relational Mapping framework providing tools and interfaces for objects persistance on relational databases. Main implementation specifications are provided in the following sections.

4.3.1 Spring-based implementation

The Spring Framework is based on a number of design patterns. For the implementation of Argo we relied mainly on two of them: i) *Inversion of Control* and ii) *Model-View-Controller*.

Inversion of Control While in traditional programming the programmer produces a custom code that expresses the purpose of the program and calls into reusable libraries, in IoC applications the purpose of the program is reusable and the programmer provides her/his custom libraries. In Argo, we relied on this programming pattern and we were able to ignore (almost) completely all the aspects related to the user management, session management and database connection management. As an example, we provided a custom library for user authentication based on the Radius protocol and associated with the credentials of the University of Studies of Milan, allowing the students of our university to log-in with their academic credentials. An extract of the code for injecting our authentication class into the Spring Framework is presented in Figure 4.1.

```
<beans:bean class="org.springframework.security.web.access.
    expression.DefaultWebSecurityExpressionHandler"/>
  <authentication-manager>

    <authentication-provider user-service-ref="
    customUserDetailService">
      <password-encoder hash="md5"/>
    </authentication-provider>

    <authentication-provider ref="radiusAuthManager">
    </authentication-provider>

  </authentication-manager>
</beans:beans>
```

Listing 4.1: Authentication library injection in the Spring Framework

Model-View-Controller. The Spring MVC Framework is based on the following components

- *Model*. This component works as an interface to the data repositories (e.g., DBMS, files, web resources). All the read/write operations needed by the ap-

plication are performed by relying on the models created to represent the data resources. In Argo, the models are represented by Hibernate objects and will be introduced in Section 4.3.2.

- *View*. This component manages the user interface. Views are invoked by the controllers to create the visual structure of the application. An example of view in Argo is the *home* view, showing some details of the user profile (see Listing 4.2).
- *Controller*. This component is the core of the Spring MVC Framework. The controller is always active and listens to the user http requests. Each controller is associated with a specific URL and performs the logics necessary to accomplish the user request. An example of controller in Argo is the HomeController: it relies on the User model to retrieve the current user information and calls the *home* view.

```
<div class="container">
  <div class="jumbotron">
    <h2>Welcome ${firstName}</h2>
    <p>You are at the ${rank.get("rank")}th place out of ${rank.get(
    "total")} total workers</p>
    <p>You have answered ${answeredTasks} tasks</p>
    <p>Your current score is ${score}</p>
    <br/>
    <p><a href="<spring:url value='/task/get'/>" class="btn btn-
    primary btn-lg">Request new task</a></p>
  </div>
</div>
```

Listing 4.2: Home view in Argo

4.3.2 Hibernate-based database access

Through the Hibernate framework, it is possible to develop persistent classes (i.e., automatically synchronized with a relational database) following traditional object-oriented patterns (e.g., inheritance, polymorphism). The developing procedure is composed by two main steps: i) definition of the Java beans (i.e., classes representing an object with all its properties) and ii) pairing of the Java bean to the relational database. After the execution of these two steps, the Hibernate Framework is able to reflect each

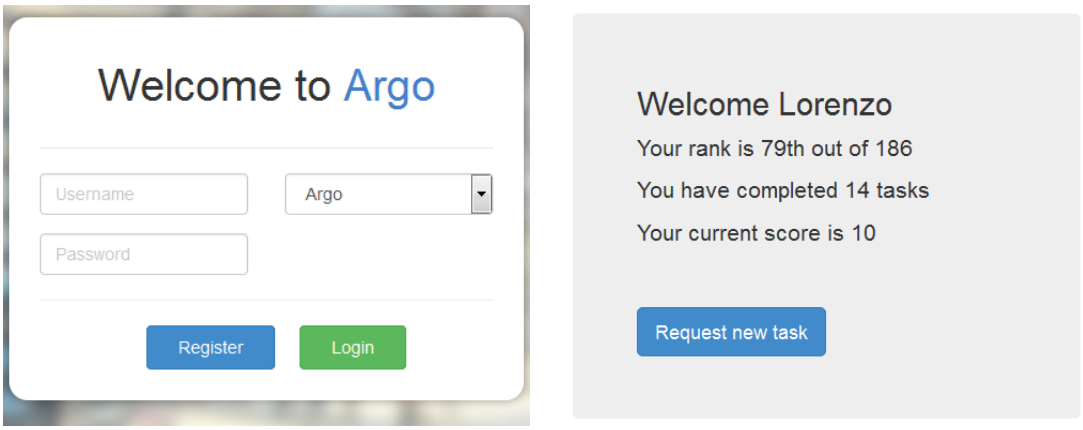
change made on the Java bean directly to the external database. In Argo, we relied on the Hibernate Framework to provide the models required for data management. An example of Model in Argo is the User class, representing all the information related to a Argo user (an extract of this class is shown in Listing 4.3).

```
@Entity
@DynamicUpdate(true)
@SelectBeforeUpdate(true)
@Table (name = "`user`")
public class User implements Serializable{

  @Id
  @GeneratedValue(strategy= GenerationType.IDENTITY)
  @Column (name = "id")
  private Integer id;

  @NotEmpty
  @Column (name = "firstname", nullable = false)
  private String firstname;

  @NotEmpty
  @Column (name = "lastname", nullable = false)
  private String lastname;

  @Column (name = "username", nullable = false)
  @NotEmpty
  private String username;

  @Column (name = "password", nullable = false)
  @NotEmpty
  private String password;

  @Column (name = "score", nullable = true, updatable=false)
  private Integer score;
```

Listing 4.3: User model in Argo

4.4 Example of task execution

The Argo prototype is a web application and can be accessed through a web-browser. After the authentication form (see Figure 4.2(a)), the Argo home-page of the user is shown (see Figure 4.2(b)).



(a) (b)

Figure 4.2: Argo prototype - login page (a) and home page (b)

The Argo home page shows to the worker his current rank, score and number of executed tasks. Through the Argo home page the worker is also able to request a new task. The task request triggers the task scheduler that assigns a new task to be executed.

Task execution. An example of a choice task execution is shown is Figure 4.3. The task execution page is structured in four main parts: i) description box, ii) context box, iii) answer box, and iv) current worker rank box. The description box shows to the worker the description d of the current task, namely a natural language text describing the activity the worker has to perform to solve the task. The context box shows to the worker the context c of the current task, that is a summary of all the information needed by the worker to perform the task. The answer box shows to the worker the possible answers PA or a text input field, depending on the task typology. Finally, the worker rank box shows to the worker her/his current rank.

If the worker is not able to provide an answer to the proposed task, she/he can reject the task: by refusing a task the worker will not receive the salary nor the award and her/his trustworthiness will remain unchanged. When a worker $\mathcal{W}$ rejects a task $\mathcal{T}$, the

task scheduler will never re-assign $\mathcal{T}$ to $\mathcal{W}$ and the worker will be able to request a new task.

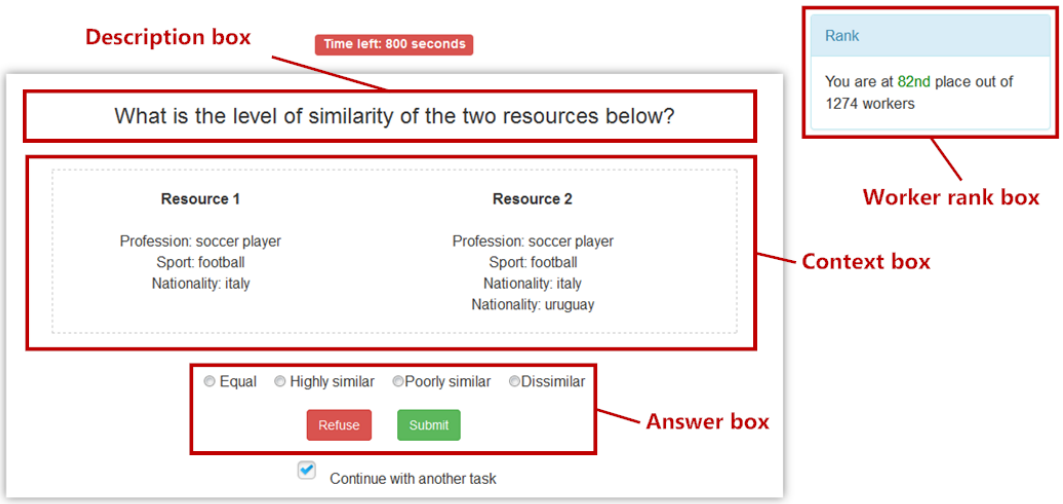


Figure 4.3: Argo prototype - example of choice task execution

User statistics. Each registered user can access to the statistics page, where summary information about executed tasks, worker score, and worker trustworthiness are shown. An example of statistics page is shown in Figure 4.4. This page is divided in two main parts: the current user statistics and the hall-of-fame.

- *Current user statistics*. In this section the user can see a summary of the work she/he has executed. More in detail, the user score and trustworthiness are presented and correlated with information about committed/uncommitted tasks.
- *Hall-of-fame*. In this section, the user is able to see the top-ten users (ranked by score) participating in the crowdsourcing activities. Each user can hide her/his personal information from the hall-of-fame by modifying her/his privacy settings.

Administration. The administrator page can be accessed only by authorized user(s). Through the administration page the user is able to set the default parameters of the Argo prototype (e.g., quorum, group size $s_{\mathcal{G}}$, standard deviation threshold th_{stddev}) and

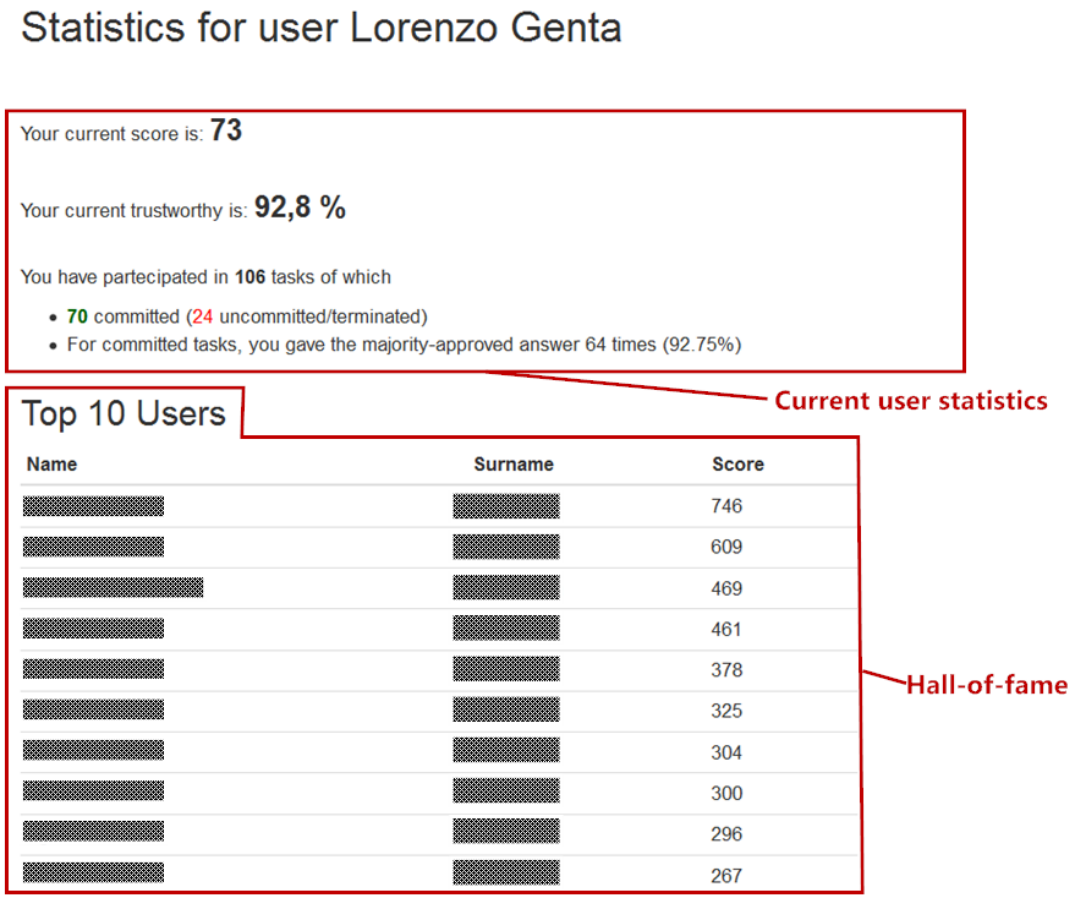


Figure 4.4: Argo prototype - statistics page

to view all global statistics about task execution, task commitment rate, and task refusal rate.

Chapter 5

Evaluation of Argo against the SQUARE benchmark

For an application-independent evaluation of Argo, we exploit the SQUARE benchmark [Sheshadri and Lease, 2013], defined for evaluation of *consensus-based* crowdsourcing systems. SQUARE provides 6 public datasets of factual tasks, each one containing a set of really-executed tasks. A task is executed by a certain number of workers (the number differs from one dataset to another), each one providing her/his own answer, called *label*. Moreover, a task is associated with an expert-validated answer, called *gold label*. The SQUARE datasets are:

- **HCB** [Buckley et al., 2010]: it contains tasks where workers have to evaluate the relevance/non-relevance of a web page as a result for an English search query. Each task is assigned to 5 workers.
- **RTE** [Snow et al., 2008]: it contains tasks where workers have to evaluate implication/non-implication of natural language sentences. Each task is assigned to 20 workers.
- **SpamCF** [Ipeirotis et al., 2010]: it contains tasks where workers have to evaluate whether a task should be considered as spam/non-spam, according to certain considered criteria. Each task is assigned to 7 workers.
- **TEMP** [Snow et al., 2008]: it contains tasks where workers have to evaluate

whether an event follows/not-follows another event. Each task is assigned to 10 workers.

- **WB** [Welinder et al., 2010]: it contains tasks where workers have to evaluate if a picture shows/not-shows a water-bird. Each task is assigned to 39 workers.
- **WVSCM** [Whitehill et al., 2009]: it contains tasks where workers have to evaluate if a person in a picture is/is-not smiling. Each task is assigned to 20 workers.

In SQUARE, the evaluation is expressed in terms of *accuracy* which measures, for a target crowdsourcing system $\bar{S}$ to be evaluated, the number of correctly-labelled tasks (i.e., the label provided by $\bar{S}$ corresponds to the gold label provided in the benchmark) over the total number of executed tasks, that is:

$$\text{Accuracy} = \frac{C}{C+E}$$

where C is the number of correctly labelled tasks, and E is the number of erroneously labelled tasks.

For evaluation of Argo against the SQUARE benchmark, we simulated an execution of Argo over the tasks of the SQUARE datasets. We composed groups with size corresponding to the number of workers provided by SQUARE for each task. For example, in the HCB dataset, each task is associated with 5 labels/answers, thus the group size in our simulation is $s_{\mathcal{G}} = 5$. The Argo simulation was configured with a maximum number of task executions $k = 1$ (i.e., the tasks are not re-assigned) since the benchmark provides only one set of worker answers for each dataset. The initial trustworthiness value is set to $\tau_0 = 0.7$ and the history weight is set to $w_h = 0.8$.

Comparison of Argo against the SQUARE benchmark. In [Sheshadri and Lease, 2013], the results of evaluating a number of different consensus-based crowdsourcing systems/techniques against the SQUARE benchmark are reported. In detail, the evaluated applications are: MV (Majority Voting) [Kumar and Lease, 2011], ZC [Demartini et al., 2012], DS [Dawid and Skene, 1979], GLAD [Whitehill et al., 2009], RY [Raykar et al., 2010]. In Table 5.1, we present the comparison of Argo against these systems/techniques. In the table, we provide detailed results for the MV mechanism, the most-commonly adopted crowdsourcing technique for consensus verification that provides the best performances [Sheshadri and Lease, 2013]. According to

	HCB	RTE	SpamCF	TEMP	WB	WVSCM	**AVG**
Argo ($q = 0.7$)	69.7	97.7	66.0	97.5	89.8	82.0	**83.8**
DS	–	–	–	–	–	–	**82.2**
CUBAM	–	–	–	–	–	–	**81.5**
RY	–	–	–	–	–	–	**80.9**
MV	64.8	91.9	66.0	93.9	75.9	72.3	**79.3**
GLAD	–	–	–	–	–	–	**78.7**
ZC	–	–	–	–	–	–	**77.2**

Table 5.1: Comparison of accuracy results for Argo against the SQUARE evaluated systems/approaches.

Table 5.1, we observe that Argo always outperforms MV in terms of accuracy, except the case of the dataset SpamCF where the two approaches achieve the same result. We observe that the Argo results should be considered as a worst-case since the tasks that remained uncommitted in the benchmark simulation would have been re-executed with other groups in normal situations, according to the LiquidCrowd approach. After re-execution, these tasks could reach commitment and the Argo accuracy could even improve.

With respect to results of the other consensus-based crowdsourcing systems presented in [Sheshadri and Lease, 2013], we point out that an analytic comparison of Argo for each dataset is not possible due to the fact that per-dataset accuracy results are only available for the MV approach. However,we can perform an aggregated comparison by considering the average accuracy of the systems results over all 6 datasets. We observe that Argo provides an average result comparable with those provided by the other crowdsourcing systems (see last column of Table 5.1). For instance the average accuracy of Argo is 83.8% while the other considered systems span from 77.2% of ZC to 82.2 of DS (see Table 5.1). It is important to note that all the evaluated systems (apart from MV) require a lot of computational effort for determining whether consensus has been reached or not due to the fact that they enforce statistical-based techniques. On this point, we highlight that Argo provides a very efficient solution since only few operations are required for a a-posteriori consensus verification according to the supermajority mechanism.

		q = 0.51		q = 0.6	q = 0.7	q = 0.8
		no bop checking (A)	bop checking (B)	bop checking	bop checking	bop checking
HCB	*Accuracy*	64.9	68.8	67.2	69.7	73.3
	Commitment	98.1	63.4	76.0	64.1	38.7
RTE	*Accuracy*	92.7	95.8	95.7	97.7	98.8
	Commitment	99.0	86.3	86.4	66.1	42.6
SpamCF	*Accuracy*	66.0	66.0	66.0	66.0	66.0
	Commitment	100.0	100.0	100.0	100.0	100.0
TEMP	*Accuracy*	94.4	96.3	96.6	97.5	95.9
	Commitment	100.0	94.4	94.2	85.3	37.2
WB	*Accuracy*	75.7	77.2	85.7	89.8	94.7
	Commitment	96.3	94.4	78.5	55.1	35.5
WVSCM	*Accuracy*	72.4	77.9	75.6	82.0	83.9
	Commitment	98.1	82.4	79.9	62.9	35.2
AVG	*Accuracy*	**77.7**	**80.3**	**81.1**	**83.8**	**85.5**
	Commitment	**98.6**	**86.8**	**85.8**	**72.3**	**48.2**
STDDEV	*Accuracy*	**11.8**	**11.9**	**12.4**	**12.5**	**12.3**
	Commitment	**1.3**	**12.0**	**8.7**	**15.4**	**23.3**

Table 5.2: Evaluation of Liquid Crowd consensus management on the SQUARE benchmark.

Argo tuning based on the SQUARE benchmark. By relying on the SQUARE benchmark, we also evaluated the performances of Argo with and without the use of bop-constraint and under different quorum values.

For this evaluation, besides accuracy already presented, we introduced *commitment* parameter defined as the percentage of committed tasks on the overall number of executed tasks, that is:

$$\text{Commitment} = \frac{N_c}{N_c + N_u}$$

where N_c is the number of committed tasks and N_u is the number of uncommitted tasks.

The evaluation results obtained are shown in Table 5.2. A first set of evaluations

have been obtained without the bop-constraint (col. A), and with the bop-constraint (col. B). With a quorum $q = 0.51$, we observe that the bop-constraint always increases the accuracy value due to the fact that bop-constraint imposes a more severe requirement for consensus verification than the basic quorum-constraint. On the other hand, we observe that the bop-constraint decreases the commitment value since reaching the consensus is more difficult and a high number of tasks remain uncommitted. As a result, we note that the bop-constraint has a positive impact on the accuracy, especially when low quorum values are set (e.g., $q = 0.51$). Indeed, with higher values of quorum (i.e., $q = 0.6$, $q = 0.7$, and $q = 0.8$), accuracy values are (more or less) the same with and without the bop-constraint. This is due to the fact that the higher is the quorum value, the more infrequent is the violation of the bop-constraint, since it is more difficult for a single worker to have a different quorum-compatible majority on different answer by only changing her/his task answer.

More in general, we note that the higher is the quorum q, the higher is the accuracy and the lower is the commitment. A high quorum value (e.g., $q = 0.8$) is recommended to ensure high quality/accuracy of crowdsourcing results. On the opposite, a low quorum value (e.g., $q = 0.51$) is suggested when the purpose is to limit the number of uncommitted tasks as much as possible, with the aim to reduce the additional costs of re-executions both in terms of time and worker revenues (i.e., salary and award). By exploiting the results of Table 5.2, we note that a quorum between $q = 0.51$ and $q = 0.6$ can be employed in common crowdsourcing situations to provide a good trade-off between accuracy and commitment results. As a further remark, we note that the impact of quorum on both accuracy and commitment is "dataset-dependent", in that there are datasets where changing the quorum strongly affects the Argo results (e.g., WB dataset), and others where changing the quorum does not produce significant effects on the Argo results (e.g., SpamCF dataset).

Chapter 6

Liquid Crowd application to web data classification and experimental evaluation

The experimental evaluation of the LiquidCrowd approach has been performed through the Argo prototype in the web-data classification context. In Section 6.1, the experimentation application cases are presented. In Sections 6.2 and 6.3, the application background and the experimentation techniques are shown. Finally, in Section 6.4 the obtained results are presented and discussed.

6.1 Presentation of the application cases

The capability to effectively browse and explore the huge amount of data actually available on the web is widely recognized [Marchionini, 2006; Bozzon et al., 2010; Castano et al., 2012a]. Solutions for web data exploration are emerging, to support non-expert users in formulating conventional keyword-based retrieval queries through high-level, concept-based information structures [Herzig and Tran, 2012]. The *linked data* [Bizer et al., 2009] paradigm introduced a new way of exposing, sharing and connecting pieces of data, information and knowledge. In [Castano et al., 2012b, 2013a], we presented a solution based on aggregation and abstraction techniques to transform a basic, flat view of a (potentially large set) of linked data, into an *in*Cloud, that is,

a high-level, thematic view enabling a more effective, topic-driven exploration of the underlying dataset. Through classification techniques, we identify thematic clusters of similar-data. Through labelling techniques, we extract suitable labels capturing the theme dealt with by a data cluster. Although the *in*Cloud construction process has been conceived as a combination of automated techniques, we argue that the *in*Cloud structure cannot be defined by an automated process only, since there are some critical issues related to similarity-based classification and labelling that would benefit from human input through crowdsourcing [Castano et al., 2013b].

6.2 Web data classification: a feature based approach

In [Castano et al., 2012b, 2013a], we introduced the notion of *in*Cloud as a high-level, intuitive organization structure capable of representing at a glance a (generally wide) collection of web resources pertaining to a given search topic/theme (see Figure 6.1). Web resources in an *in*Cloud come from multiple webs (i.e., flat web pages, linked-data web, semantic web) to provide a comprehensive picture of all available information, both objective and subjective. In fact, an *in*Cloud pervasively collects together objective information produced by conventional web data resources and subjective information derived from social web resources. In such a way, the official information about the target entity that is usually provided by web sites and broadcasters is complemented with the so-called user generated content as it can be derived from microblogging and other similar kinds of information sources.

In particular, *in*Clouds aim at enforcing the following goals.

Conceptual web data description. An *in*Cloud provides a concept-based view of an underlying set of web resources through aggregation and abstraction techniques. An *aggregation-by-similarity* is first performed to cluster data that are somehow related in the same group. This way, similar data, like people that have the same profession, are grouped together in a single cluster. Then, *abstraction-by-labels* is executed to associate a cluster with a synthetic and representative description of the data therein contained. This way, a label, like singer, can be associated with a cluster to conceptualize the fact that the cluster contains persons that are singers.

Smart web data exploration. An *in*Cloud supports the user in effectively brows-

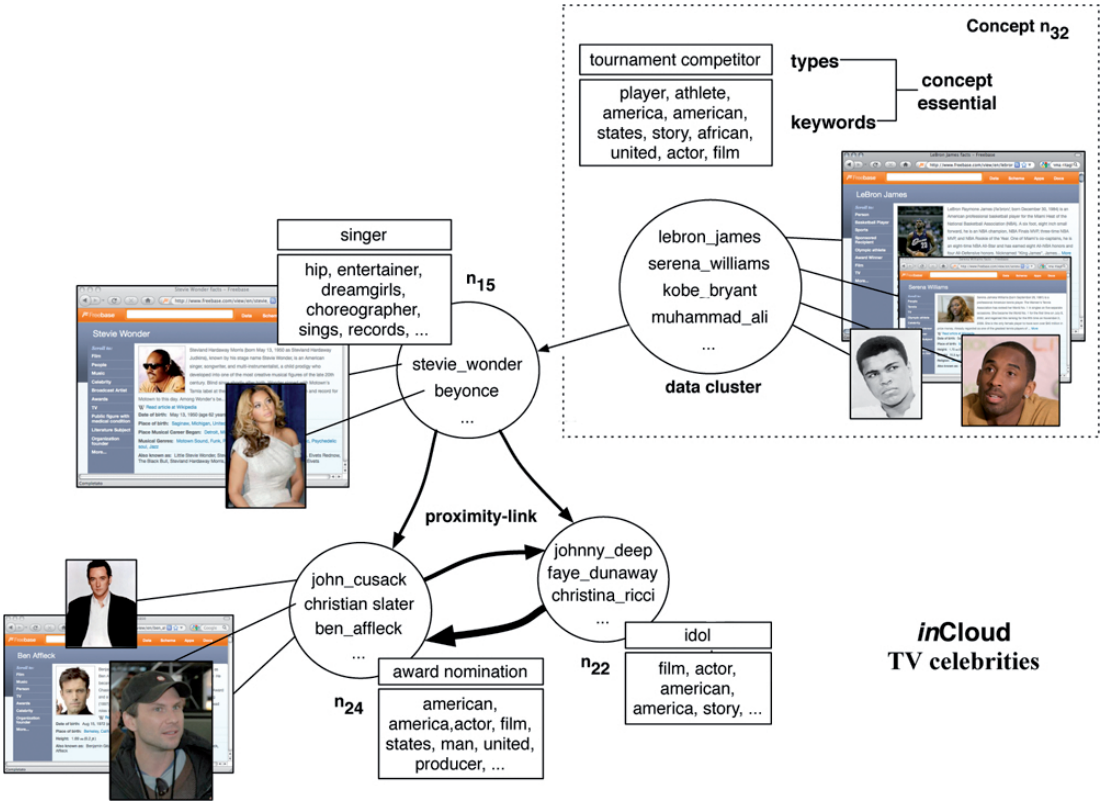


Figure 6.1: An example of a portion of *in*Cloud

ing the underlying set of web resources according to her/his preferences. To visually represent the importance of a concept/cluster within an *in*Cloud, a *ranking-by-prominence* strategy is enforced. This way, the size of a cluster is set to capture the relative importance/popularity of its data with respect to the data belonging to the whole *in*Cloud. The higher the importance of the cluster contents for the *in*Cloud, the higher the size of the cluster to better focus the user attention on such a cluster. In addition, a *link-by-proximity* strategy is pursued to interlink pairs of clusters with the aim to suggest possible exploration paths across similar/close concepts of the *in*Cloud. This way, it is possible to enable a user to browse from one cluster to another according to a similarity-based or closeness-based criterion. For instance, a proximity link between the cluster singer and the cluster award nomination captures the fact that their corresponding concepts are close and thus such a link is a potentially-interesting exploration path to follow.

6.2.1 The *in*Cloud construction process

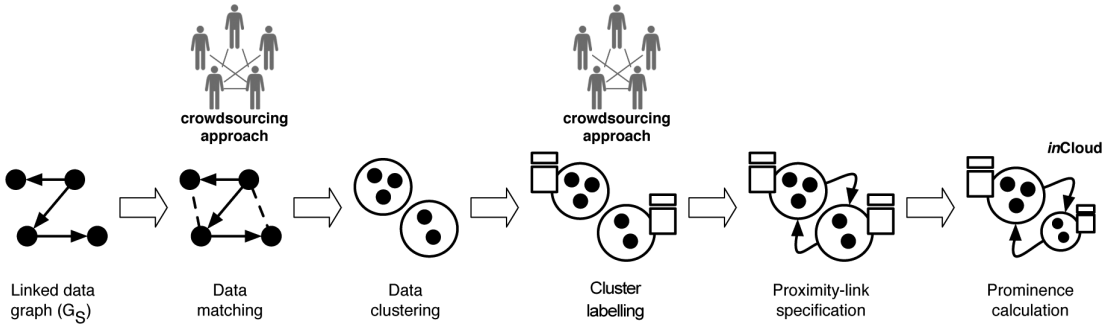


Figure 6.2: The *in*Cloud construction process

Starting from a set $\mathcal{S}$ of web resources pertaining to a certain topic/theme of interest, the process of *in*Cloud construction is articulated in two main phases, namely *data classification* and *in*Cloud *data abstraction* (see Figure 6.2). The phase of linked data classification is enforced to transform the initial set $\mathcal{S}$ of flat web resources into a set of similarity-based clusters *CL* [Ferrara et al., 2013, 2014b]. To this end, a *data matching* step is first executed to determine all the pairs of similar resources in the set $\mathcal{S}$. The proposed matching techniques, evaluate not only a degree of similarity between data but also provide information about the causes of the similarity and the reliability of the matching results. Then, a *data clustering* step is performed to build clusters out of the similarity values discovered through matching. The proposed clustering algorithm is based on a hierarchical clustering technique of agglomerative type for taking into account data features as a first-class citizen in the aggregation process. After classification, the phase of *in*Cloud abstraction is enforced to transform the set of clusters *CL* into an *in*Cloud [Castano et al., 2013a]. First, a *cluster labelling* step is executed to define the set of concepts N by equipping each cluster cl_i with its corresponding labels d_i describing cluster contents at a conceptual level through a set of relevant terms. Then, a *proximity-link specification* step is performed to define the set E of proximity links connecting the pairs of similar concepts of N. For each pair of concepts $n_i, n_j \in N$, a proximity degree x_{ij} is calculated. Finally, a *prominence calculation* step is performed to complete the *in*Cloud creation by determining the prominence p_i for each concept n_i.

In applying the process of Figure 6.2, we note that two critical issues in the *in*Cloud

creation are related to data matching and cluster labelling, in that these steps cannot be only based on the output of an automated procedure. The idea is to employ crowdsourcing to bring human expertise and domain knowledge in the construction process of Figure 6.2 with the aim to improve the quality and the effectiveness of the resulting *in*Cloud clusters and labels.

6.3 LiquidCrowd for web data classification

The application of Argo to the *in*Cloud construction process has been performed on the steps marked in Figure 6.2. In the following sections, we introduce the application methodology for both similarity evaluation and cluster labelling.

6.3.1 Similarity evaluation

Similarity evaluation is a well-known research topic with applications to a number of state-of-the-art fields about data management spanning from data integration [Halevy et al., 2006] to instance matching and ontology alignment [Castano et al., 2011]. In general terms, we define similarity evaluation as the problem of automatically assessing the degree of similarity between two given resources based on their available descriptions. For the application of LiquidCrowd, we consider web-of-data resources which are instance/individual descriptions associated with a set of *feature-value* pairs, each one denoting a specific property of the considered resource [Ferrara et al., 2014a]. A commonly adopted matching technique to evaluate the similarity degree of two resource descriptions is based on counting the number of common feature-value pairs over the total number of feature descriptions. A high similarity degree is obtained when the two instance descriptions are equal (i.e., they share all feature-value pairs) or highly similar; a low similarity degree is obtained when resources have a few commonalities or even completely different descriptions [Ferrara et al., 2014a].

Consider the following simple web-of-data descriptions of Alessandro Del Piero and Edison Cavani extracted from the Freebase repository[1].

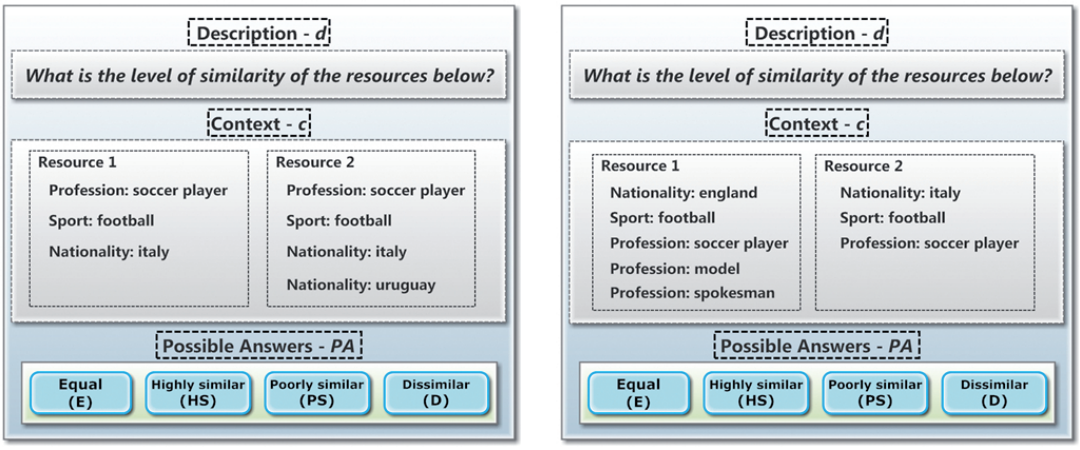
[1] http://www.freebase.com

alessandro del piero	**edison cavani**
profession: soccer player	profession: soccer player
sport: football	sport: football
nationality: italy	nationality: italy
	nationality: uruguay

By comparing these two descriptions based on their feature-values, we note that Alessandro Del Piero and Edison Cavani have high similarity degree as they share three feature-value pairs. A crowdsourcing approach can be effectively employed to perform similarity evaluation over web-of-data resources. A worker can evaluate the similarity between two resources by observing their descriptions at-a-glance and by assigning a certain degree of similarity based on her/his background knowledge and problem-understanding capabilities/attitudes. Similarity evaluation falls in the category of collaborative tasks, since a distinction between right and wrong answers is not possible/nor suitable and the goal of the crowdsourcing task is to capture the crowd feeling about the perceived similarity among the considered resources.

Similarity evaluation has been modelled as a choice-task in LiquidCrowd. This way, we aim at pursuing two opposite goals at the same time. On one hand, the use of a predefined set of possible answers gives a certain level of expressiveness to the worker in answering to the given task. On the other hand, the limited number of answers enables Argo to supervise the answer dispersion and to simplify the answer composition within groups [Krosnick and Fabrigar, 1997]. For similarity evaluation, the task structure S is shown in Figure 6.3.

We stress that, in visualizing the resources to compare, a *blind* presentation has been adopted, in that only feature-value pairs are shown, while specific names or pictures of the individual have been discarded. This way, in assessing the similarity degree the worker can only rely on the available features and she/he is not influenced by knowing who the specific individual is. For example, considering the task example of Figure 6.3(a), a worker has to evaluate level of similarity of these two individuals without knowing that they are Alessandro Del Piero and Edison Cavani but solely on the basis of the fact that i) they are both soccer/football players and ii) the individual on the left is Italian, while the individual on the right is both Italian and Uruguayan.



(a) **(b)**

Figure 6.3: Examples of choice tasks

6.3.2 Cluster labelling

The need of finding one or more label(s) to suitably denote real-world object conceptualizations arises in a number of different fields of computer science. In conceptual database design, finding the appropriate names for the entities in the conceptual schema is a key requirement to guarantee that the resulting schema is expressive to capture the meaning of the objects of the real-world that are abstracted as schema entities [Batini et al., 1992]. In object-oriented software design, appropriate class names are selected to express the meaning and characteristics of the objects the class represents [Butler et al., 2011]. In ontology engineering, sets of individuals are classified in form of concepts whose name is a term reflecting the meaning of the concept to be agreed by the target community [Svátek et al., 2009]. In tagging systems, a set of tags are chosen by users to suitably describe the content of web resources (e.g., text documents, images, videos, web pages) to be shared by the social network community.

We define a cluster as a collection of web resources characterized by common properties, namely $C = \{r1, r2, ..., rn\}$. A property p of a resource r represents an elementary attribute or feature of r and it is formalized as a pair <propertyName-value>.

We call cluster labelling the process of choosing either the name for C, that is, a

unique label denoting C, or the tag-set for C, that is, a set of labels annotating C. In both cases, name labels or tag labels must be appropriate for summarizing the cluster contents, namely web resources and their properties.

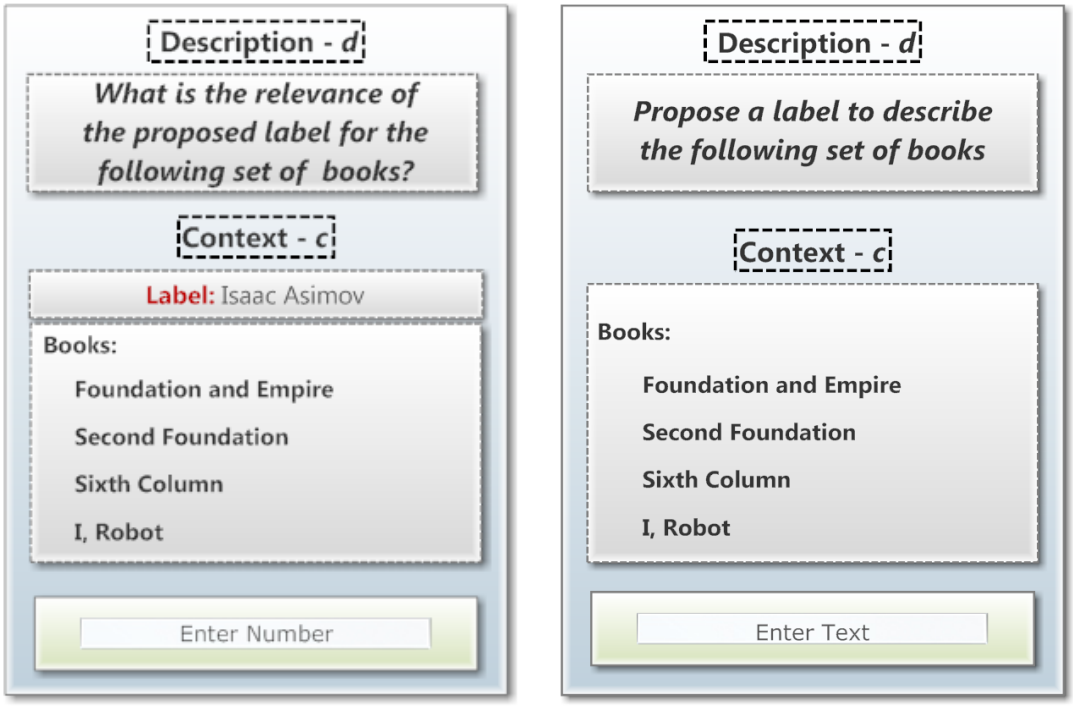
Consider for example an *in*Cloud related to the topic "books". In this context, consider the cluster $C = \{$Foundation and Empire, Second Foundation, Sixth Column, I, Robot$\}$.

All resources in C are books with topic *Science Fiction* and author *Isaac Asimov*. Consider the following three labels, obtained exploiting the knowledge we have on C: 1) *Book*, 2) *Isaac Asimov*, and 3) *Science fiction*. We note that the label 1) is good at describing the resources of the cluster C but it is not a peculiar characteristic of C because it is exactly the topic of the considered *in*Cloud (i.e., each cluster in the considered *in*Cloud could be described by the label *Book*). Thus, the best labelling for C would derive from the cluster topic (i.e., science fiction) and from the cluster author (i.e., Isaac Asimov).

Addressing the labelling problem through automated techniques is difficult, because label suitability requires the capability of understanding entity contents and label meaning. To comply with these requirements, we employ a crowdsourcing-based approach for entity labelling where the selection of the appropriate label(s) is managed as a collective task based on LiquidCrowd mechanisms for consensus evaluation. This way, we are able to combine the information about labels extracted from object properties with the common sense of the workers as it emerges from the consensus reached around the labels.

The experimentation on cluster labelling has been performed with a twofold aim: i) to assess the human-perceived relevance of automatically extracted labels and ii) to gather new labels produced by the crowd.

Relevance assessment has been modelled as a score-task where workers are asked to assign a value in range $[0, 10]$ to an automatically-extracted label (see Figure 6.4(b)). The context of the score task shows to the worker the titles of the books contained in the considered cluster and the label that has to be evaluated. An example of range task execution is presented in Figure 6.5(a).



(a) **(b)**

Figure 6.4: Examples of (a) range task and (b) proposition task

New label gathering has been modelled as a proposition-task (see Figure 6.4(b)) where workers are asked to propose a label being suitable for describing the cluster contents. The context of the proposition task shows to the worker only the titles of the books contained in the considered cluster. An example of proposition task execution is presented in Figure 6.5(a).

Both in range and proposition tasks, workers should understand which are the common properties in order to assess the relevance of the proposed label or to propose a new label. We note that, in order to provide a good answer, it is very important that each worker could have the same knowledge about the proposed books. For this reason, in the context c of both range and proposition tasks, each book-title is linked to a Wikipedia page where a detailed description of the book is provided.

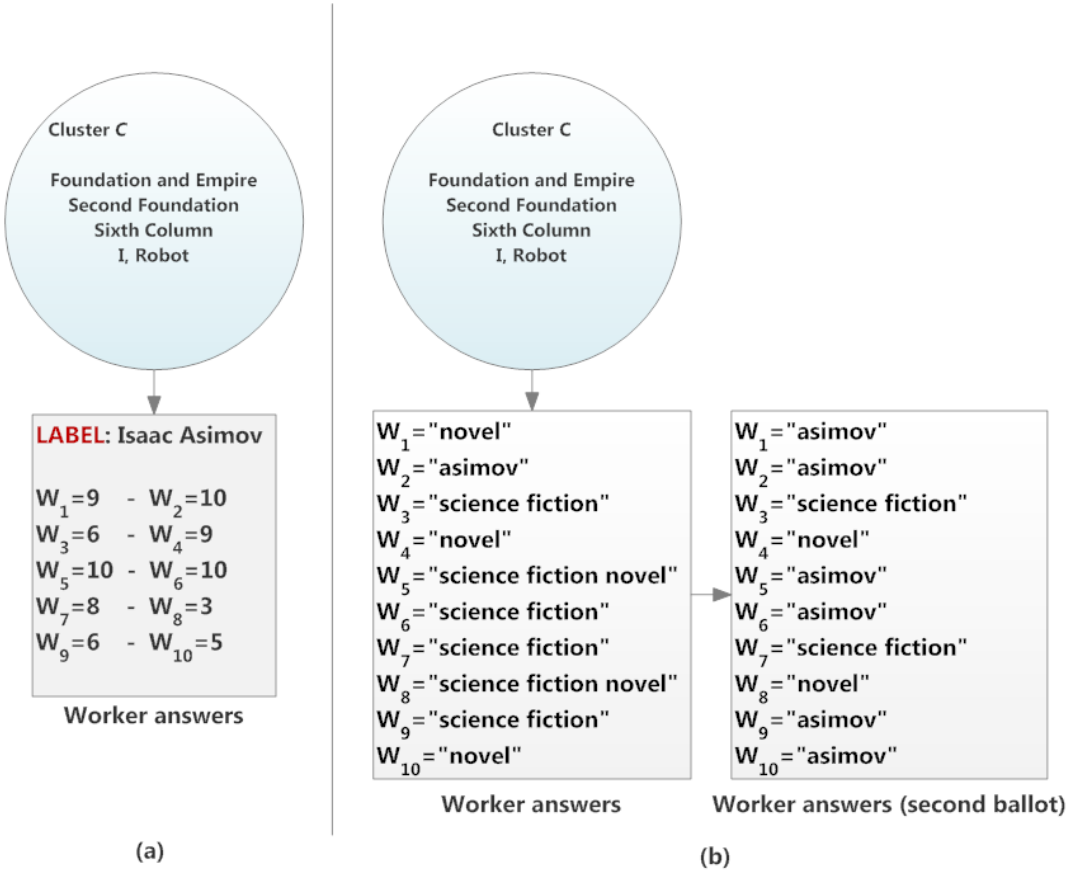


Figure 6.5: Examples of (a) range task execution and (b) proposition task execution

6.4 Experimental results and case-study-oriented evaluation

In this section, we present and critically analyse the results obtained from the first and the second application cases.

Argo configuration. The configuration of Argo adopted in the following test cases is the result of an extensive experimentation performed for tuning the prototype under different setups (e.g., different group size, different salary/award for workers, different values of trustworthiness threshold). The experimentation was based on a dataset of *general knowledge* tasks (i.e., tasks having a correct answer known a-priori) executed by 317 workers selected in a class of humanities students. As a result, we setup the

following configuration:

- quorum value $q = 0.51$;
- history weight $w_h = 0.8$;
- trustworthiness threshold $th_\tau^1 = 0$;
- each time a task is re-executed, the trustworthiness threshold is 10% increased (i.e., $th_\tau^{i+1} = th_\tau^i + 0.1$);
- the worker salary is $s = 0.1$ and the worker award is $a = 1$;

The remaining values of initial trustworthiness τ_0, group size $s_\mathcal{G}$ and maximum number of task re-executions k are differently set in the considered case studies according to the specific experimental goal.

6.4.1 Similarity evaluation test case

The case study sim-eval for web-resources similarity evaluation (see Section 6.3.1) is composed by 58 instances describing movie and sports celebrities extracted from the Freebase repository, with a total number of 275 feature-value pairs.

Argo configuration. For application to the sim-eval case study, the Argo prototype has been configured as follows:

- initial trustworthiness value $\tau_0 = 1.0$;
- group size $s_\mathcal{G} = 5$;
- maximum number of task executions $k = 3$ times;

In this case study, we set a high initial trustworthiness value τ_0 and a small group size $s_\mathcal{G}$ because of the high reliability and base-knowledge of the involved workers. Considering the small number of workers involved in this case study, we also decided to set a small value for the maximum number of task executions k in order to help the crowd executing all the provided tasks without a re-execution flooding.

The goals of this experimentation are to i) inspect the applicability of the proposed approach to a real-case study by analysing the capability of the workers in reaching the consensus, and ii) evaluate the behaviour of worker score and trustworthiness under different quorum settings in order to assess the effectiveness of the trustworthiness measure to identify different worker categories.

Experimental results. The experimentation was conducted with a crowd of 82 workers selected in a class of computer science students playing the role of workers. They are skilled in computer-science topics, with only basic notions of the web-of-data. We defined a task for each possible pair of different resources in the sim-eval case study, resulting in a total number of 1653 tasks. The experiment duration was 7 days, workers committed 1136 tasks out of 1204 (94%), with the following consensus results: $E = 15, HS = 151, PS = 317, D = 653$. The average workload (i.e., the number of tasks executed by each worker) was 84.12. With respect to objective i), the high commitment value obtained by workers in this experiment is a proof of successfully application.

Figure 6.6 shows that the worker score proportionally increases with the workload: this is due to the fact that the higher is the number of tasks a worker $\mathcal{W}$ executes, the higher is the salary received and the higher is the likelihood that $\mathcal{W}$ receives an award. We also point out that the probability to receive an award is higher when the q value is low, since a lower number of workers that have to agree on the same answer is required.

Figure 6.7 shows that trustworthiness values of workers with low workload span over the whole range of possible trustworthiness values. Moreover, we note that workers executing a high number of tasks always have high trustworthiness values, and workers with high trustworthiness values usually execute a high number of tasks. In our view, a possible explanation of this behaviour is that workers with low trustworthiness values get frustrated and stop participating while workers with high trustworthiness values get motivated and further participate to the crowdsourcing activities for increasing their score. With respect to objective ii), this is a demonstration of the effectiveness of the trustworthiness measure in identifying the capability of the workers to be in agreement with the majority. As a further remark, we note that high values of the quorum q imply high trustworthiness values (Figure 6.7). To better understand this result, consider the peculiar case $q = 1$. In this situation, a task is committed iff all

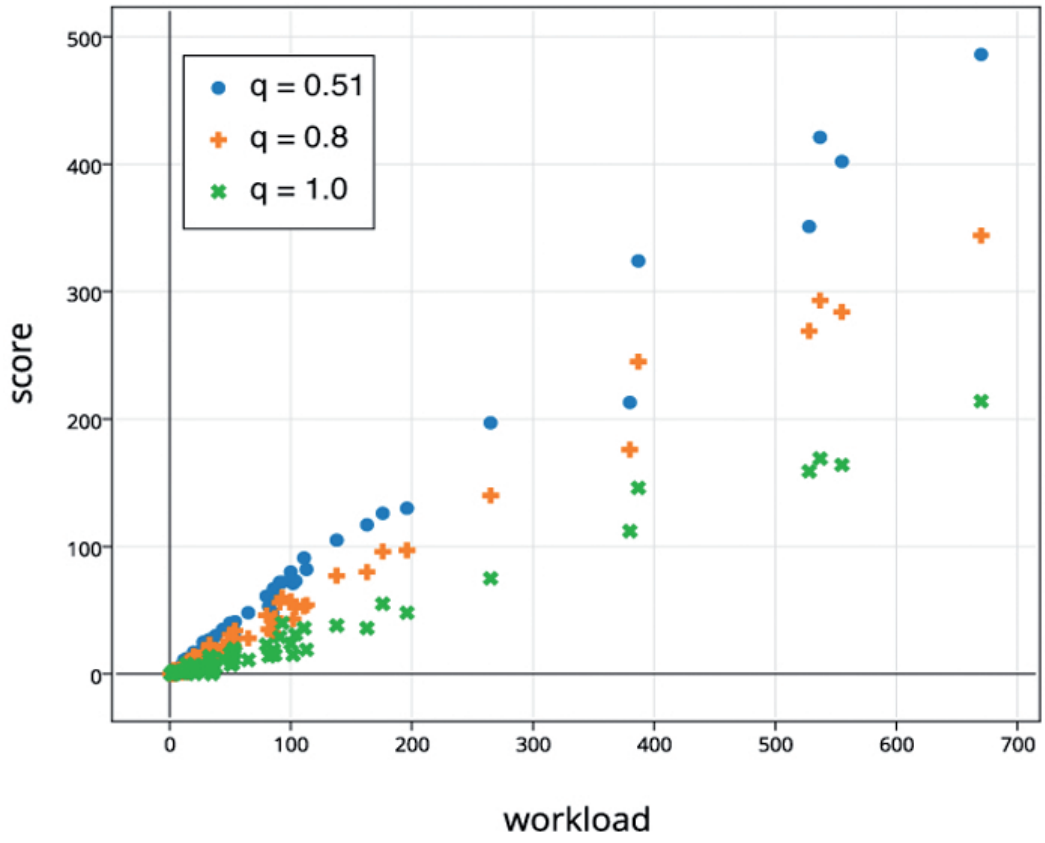


Figure 6.6: Relationship between workload and score for the case study sim-eval

the workers involved in the task execution agree on the same answer. This means that a worker can not participate to a committed task without receiving an award. Thus, it is impossible to participate to a committed task without increasing the trustworthiness value.

For a more detailed analysis of the worker behaviour, Figure 6.8 shows the correlation among score, trustworthiness and workload with $q = 0.51$. We observe that most of the workers have a trustworthiness value higher than the quorum, meaning that workers commit most of the executed tasks and the consensus obtained on the 1st-candidate-answer is usually higher than the considered quorum $q = 0.51$.

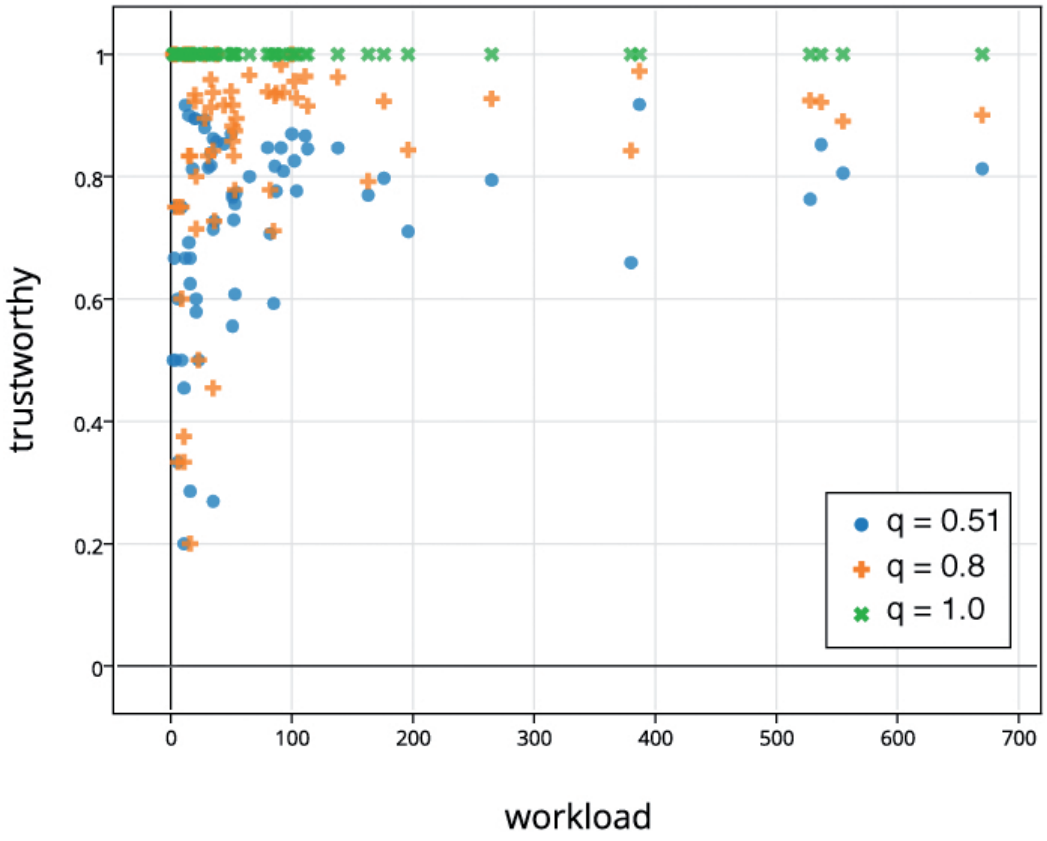


Figure 6.7: Relationship between workload and trustworthiness of workers for the case study sim-eval

6.4.2 Cluster labelling

The case study clu-lab for cluster labelling (see Section 6.3.2) is composed by 317 instances describing books extracted from the DBPedia repository, with a total number of 3139 feature-value pairs. The extracted resources have been clustered and labelled through the *in*Cloud generation process obtaining 63 clusters associated with 661 labels.

Argo configuration. For application to the clu-lab case study, the Argo prototype has been configured as follows:

- initial trustworthiness value $\tau_0 = 0.7$;

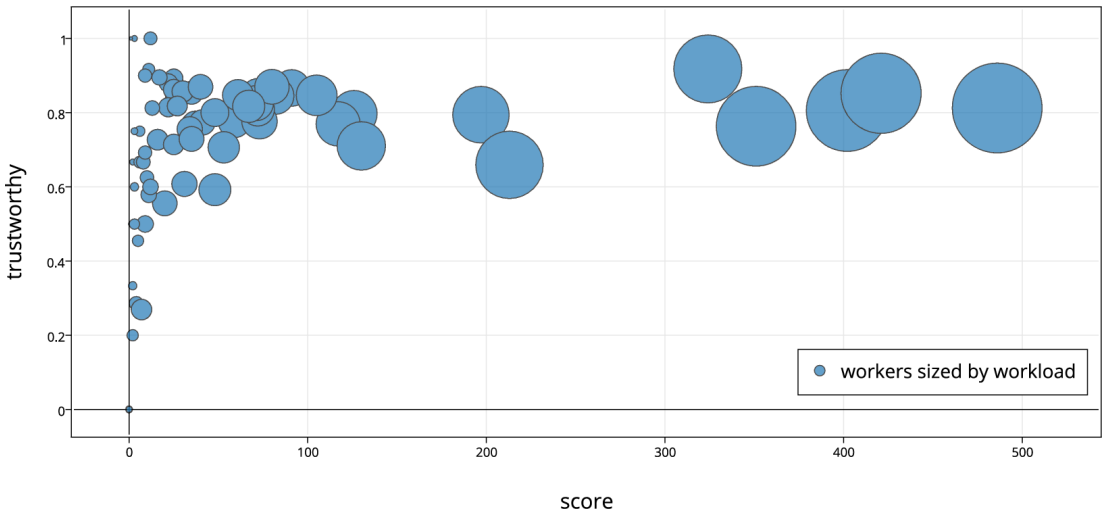


Figure 6.8: Relation between workers workload, score, and trustworthiness for the case study sim-eval

- group size $s_{\mathcal{G}} = 10$;
- maximum number of task executions $k = 5$ times;

In this case study, we set a lower initial trustworthiness value τ_0 and a larger group size $s_{\mathcal{G}}$ because of the lower reliability and base-knowledge of the involved workers. Considering the high number of workers potentially involved in this case study, we also decided to set a high value for the maximum number of task executions k in order to increase the likelihood that tasks reach commitment.

The goals of this experimentation are: i) evaluate the behaviour of worker score and trustworthiness under different quorum settings to confirm the results presented in Section 6.4.1, and ii) analyse the labels obtained through the range and proposition tasks in order to observe the behaviour of workers in proposing labels and, more precisely, their capability to identify the features that are common to all the resources in the considered cluster.

Experimental results. The experimentation was conducted with a crowd of 510 workers selected in a class of humanities students playing the role of workers. We defined three tasks with typology $t = \textit{proposition}$ for each cluster and one task with typology

$t = range$ for each label resulting in a total number of 850 tasks. The experiment duration was 20 days, workers committed 503 tasks out of 609 executed (82,6%).

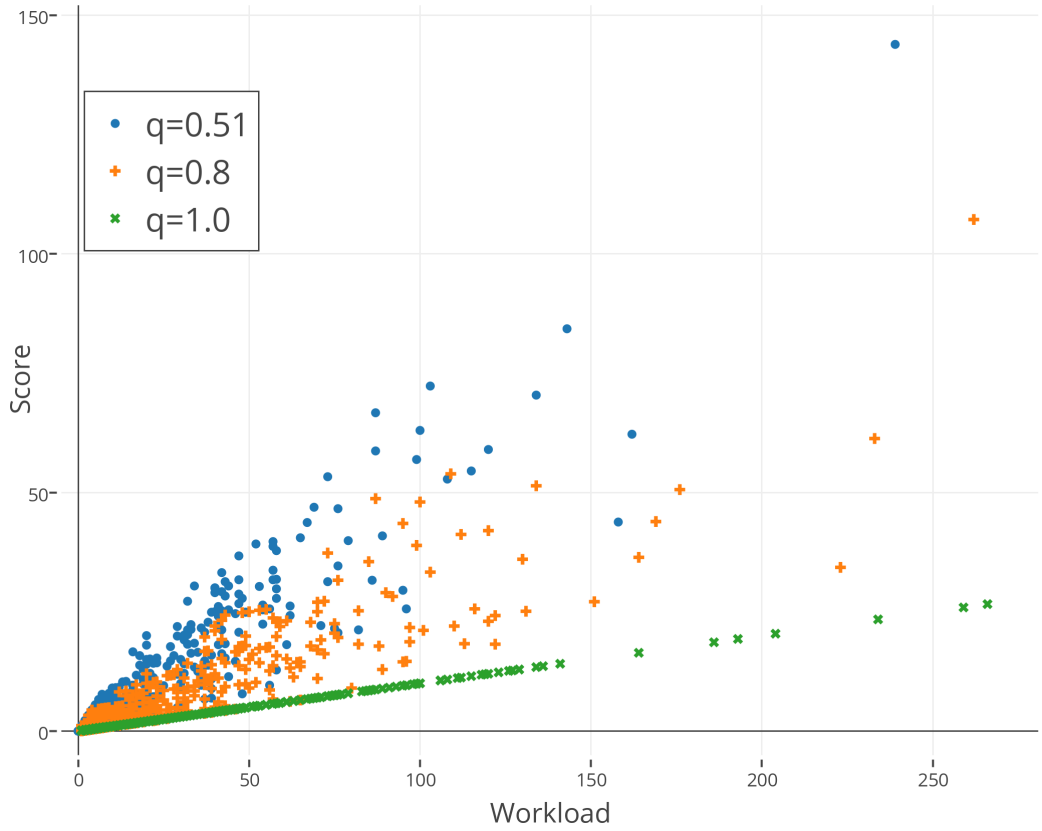


Figure 6.9: Relationship between workload and score for the case study clu-lab

With respect to the objective i), the results of the clu-lab case study confirm the results of the sim-eval case study. As we can see in Figure 6.10, the trustworthiness of the workers are distributed over a wide range of possible values, distinguishing workers capable to be part of the consensus in most of the cases (upper part of Figure 6.10) from workers that are often in contrast with the most shared opinion (lower part of Figure 6.10). We also note two differences with the sim-eval case study:

- *Workload vs. Score* (see Figure 6.9). We note that with a quorum $q = 1.0$, the score of the workers has a linear behaviour. This is due to the fact that, with the given threshold $q = 1.0$ none of the groups reaches the consensus (i.e., it never

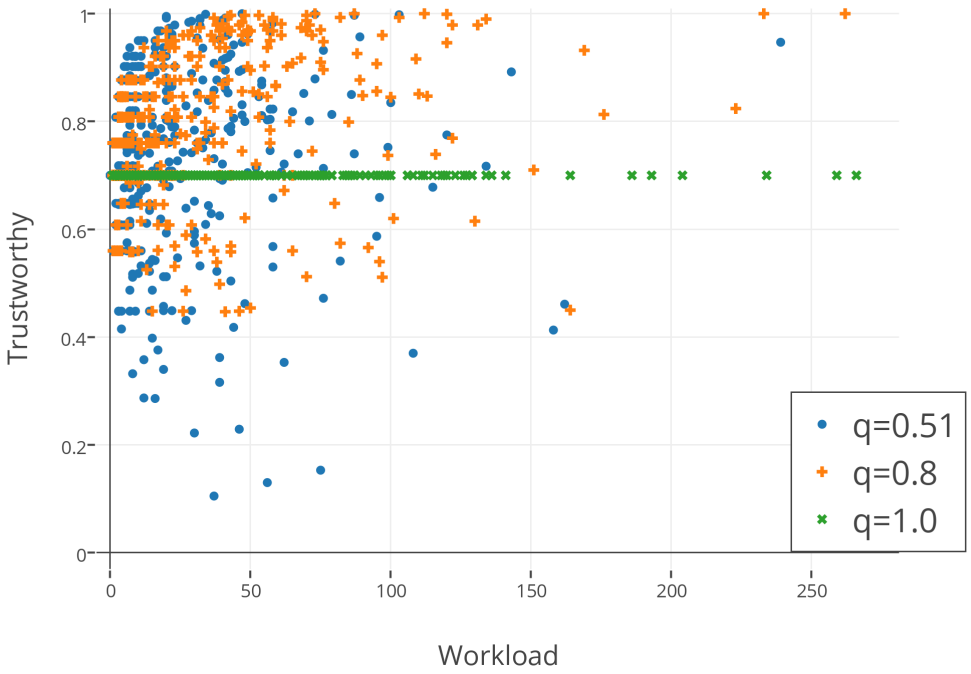


Figure 6.10: Relationship between workload and trustworthiness of workers for the case study clu-lab

happened that all the workers in a group agreed on the same answer). Thus, the workers have been rewarded only with the salary for each executed task. This is an expected result, due to the higher degree of freedom of the workers in tasks with typology $t = range$ or $t = proposition$ and to the higher group size (i.e. $s_g = 10$). For these reasons, reaching the consensus is more difficult than in the sim-eval case study.

- *Workload vs. Trustworthiness* (see Figure 6.10). We note that with a quorum $q = 1.0$, the trustworthiness values of the workers are always equal to 0.7 that is the initial worker trustworthiness ($\tau_0 = 0.7$). Since none of the groups is able to reach the consensus with quorum $q = 1.0$, the trustworthiness values never change.

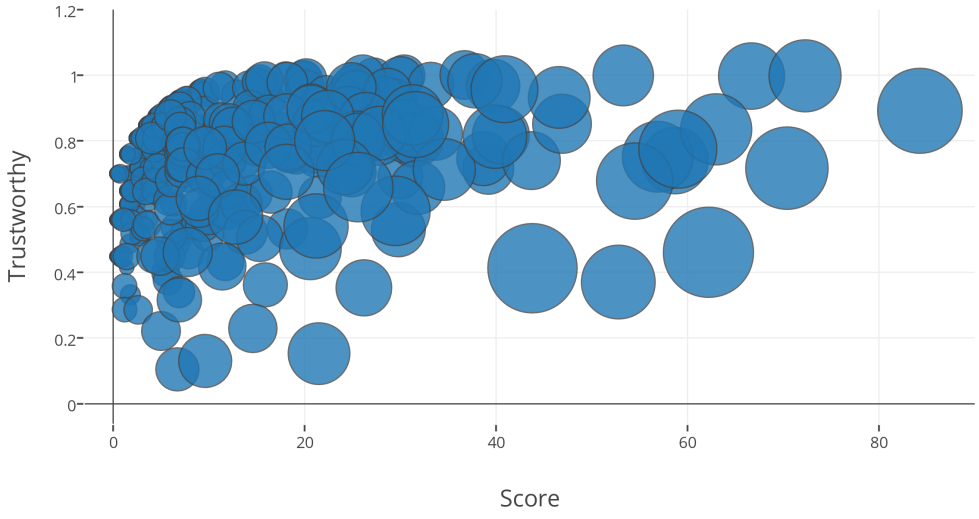


Figure 6.11: Relation between workers workload, score, and trustworthiness for the case study clu-lab

In Figures 6.12 and 6.13, we present two clusters associated with the labels produced by i) automatic techniques, ii) range tasks, and iii) proposition tasks[2], and we analyse the obtained results.

Automatic techniques vs. range tasks. In Figures 6.12(a)(b) and 6.13(a)(b), the labels obtained through automatic techniques and range tasks are presented, sorted by automatically-computed relevance (based on TF-IDF) and by crowd-computed relevance, respectively. We want to point out that the labels produced by automatic techniques and the labels produced by range tasks are identical but presented in a different order. This is due to the definition of range task for this case study, given in Section 6.3.2. With a more accurate analysis, we note that, for the cluster of Figure 6.12, automatic techniques identified doubleday mystery as the most relevant label and isaac asimov as the least relevant label. We note that the result produced by the crowd through range tasks is inverted: isaac asimov has been evaluated as the most relevant label while

[2]The full results of this case study are available at `http://islab.di.unimi.it/liquid_crowd`

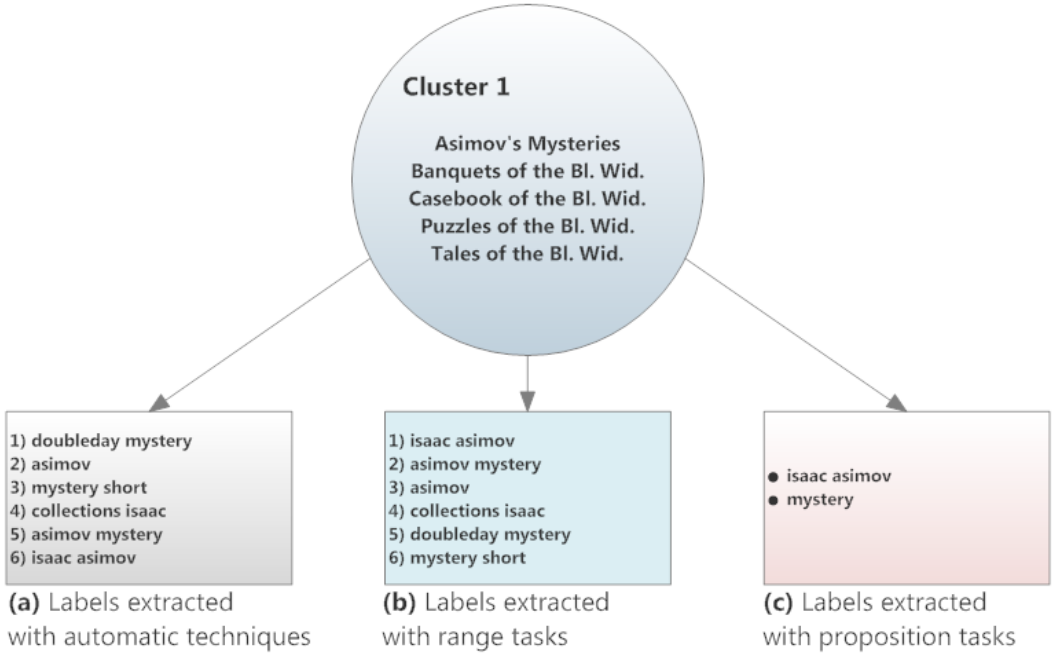


Figure 6.12: Labelling results (1)

doubleday mystery has been evaluated as one of the least relevant labels. The same happens in Figure 6.13 for doubleday and science fiction.

This means that crowd workers completely disagreed with label ranking obtained through TF-IDF techniques. This is probably due to the fact that rarest words, such as Dubleday, that are considered more important in TF-IDF techniques, could be less characterizing than more frequent words, such as Isaac Asimov or Science Fiction.

Range tasks vs. proposition tasks. In the examples presented in Figure 6.12(c) and 6.12(c), only two labels are presented. This is due to the fact that the crowd completed only two of the three produced proposition tasks. We note that the labels proposed by the crowd are almost the same of the first two labels obtained through range tasks. In Figure 6.13, we note the same situation, apart from the fact that the keyword science fiction 1900 is a partial repetition of the keyword science fiction with additional information related to the publication period. This repetition is due to the fact that the proposition tasks produced for each cluster are completely independent: this means that the workers do not have any knowledge about the already proposed

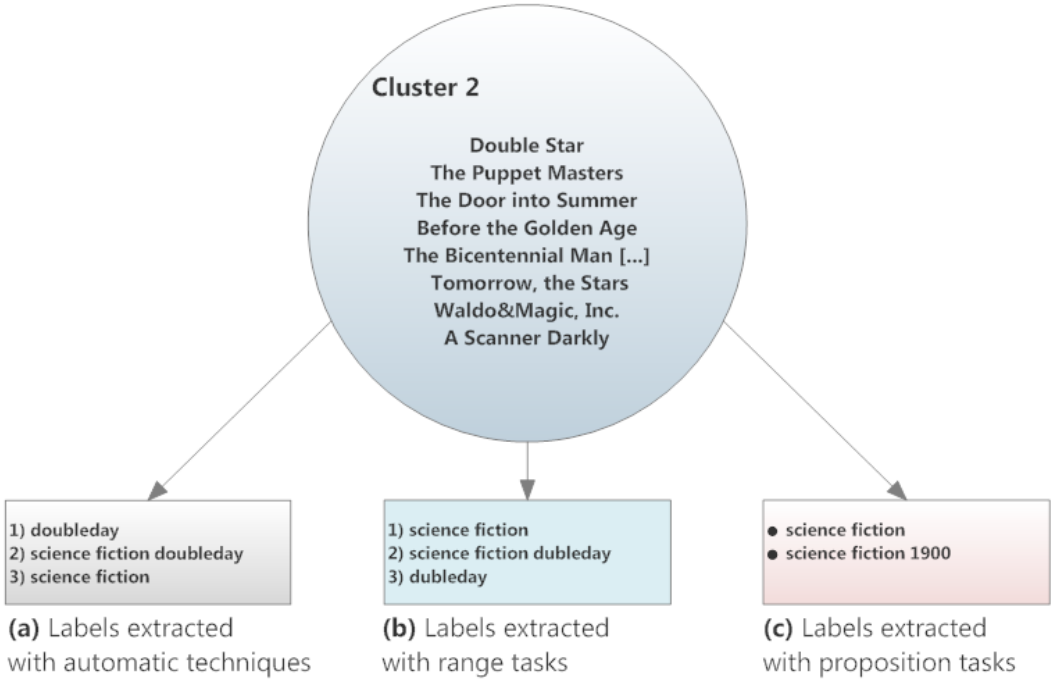


Figure 6.13: Labelling results (2)

labels, thus it could happen that they produce two ore more identical labels. As a final remark, we note that through proposition tasks we obtained the information about the publication century (i.e., 1900), that is missing in the data extracted from the DBPedia repository.

In Section 6.2.1, we introduced the adopted feature-based clustering algorithm [Ferrara et al., 2013, 2014b]. Through this clustering approach, we produced clusters where all the contained resources share a set of identical feature-value pairs that are used to characterize the cluster. For example, the cluster presented in Figure 6.12, has the following characterizing features: Publisher: Dubleday, Genre: Mystery, Author: Isaac Asimov. By performing an analysis over all the results obtained from proposition tasks, we note that, for each proposition task, at least one of the produced labels has the same meaning of one of the characterizing feature-value pairs of the considered cluster. With respect to the objective ii) of this experimentation, this means that the crowd was able to identify the reasons that brought the clustering algorithm to aggregate the provided

resources. From our point of view, this is an interesting result and can be exploited in future work to automatically produce human-comprehensible cluster labels based on the characterizing features of the obtained clusters.

Chapter 7

Conclusions and future work

In the thesis, we presented the LiquidCrowd approach and the Argo prototype for consensus-based crowdsourcing and quality assessment. The thesis focus is about i) the use of trustworthiness in addition to score mechanisms to recognize (and selectively reward) accurate workers in the execution of crowdsourcing activities, ii) the use of groups instead of single workers for task execution and subsequent quality assessment of results through consensus verification, and iii) the definition of a crowdsourcing approach capable of managing different task typologies at the same time.

The results obtained by comparing Argo with the SQUARE benchmark are promising and demonstrate the flexibility of the proposed approach over the consensus verification and trustworthiness assessment. The real-case studies demonstrate that the LiquidCrowd approach is also valuable for real application/problem solution.

7.1 Future work

We are interested in exploring the potential of LiquidCrowd extensions in the following directions.

Context-based worker trustworthiness. In LiquidCrowd, the trustworthiness of a worker is an overall value that represents her/his ability in providing consensual contri-

butions. This is a very general model that can be extended starting from the following consideration: in real applications, each worker could have different skills in different contexts (e.g., a worker could be expert in topics related to mathematics but could have no knowledge about history or politics). The proposed trustworthiness model could be improved considering not only the *overall* ability of the worker but also her/his specific ability over different contexts. A first step in this direction could be to differentiate the worker trustworthiness depending on the task typology (e.g., the worker $\mathcal{W}$ will be associated with three trustworthiness values: $\tau_{\mathcal{W}}^{choice}$, $\tau_{\mathcal{W}}^{range}$, and $\tau_{\mathcal{W}}^{proposition}$). A further extension could be to associate each task with a topic of interest and evaluate the worker trustworthiness relatively to the considered topic. This also goes in the direction proposed in [Roy et al., 2013] with the paradigm "who knows what".

Candidate answer identification for range tasks. In the proposed approach, we rely on a technique for candidate answer identification that is different for each task typology. For range tasks, we proposed a technique that identifies the arithmetic mean over all the worker contributions and that uses this value to find the closest consensus group (see Section 3.3.2). This way, we rely on the fact that workers can form only two consensus groups: one group near to the "most shareable answer" and the other one which includes the outliers. Through this technique, we do not consider the possibility to form three or more consensus groups, such as, for example, when two or more beliefs arise from worker answers. In such a situation, we need a technique capable of identifying the "centroids" of the support groups. We plan to inspect possible solutions starting from work performed in other research areas, such as modal identification [Brincker et al., 2000], unsupervised clustering [Xu and Wunsch II, 2005], and outliers detection [Hodge and Austin, 2004].

Refinement of proposition-task management techniques. The proposed approach for proposition-task management is to be intended as general demonstration of the applicability of the LiquidCrowd approach to both *decide* and *create* tasks. However, the management of proposition tasks through crowdsourcing represents an actual and challenging topic of research. In particular, we plan to work on the following open issues: i) extension of the proposition task to make it more compliant with long texts and, ii) exploration of similarity techniques to recognize equivalence among words even if the syntactic structure is different.

Other task typologies. The LiquidCrowd approach has been structured to be extensible with respect to the supported task typologies. We provided three task typologies (i.e., choice, range, and proposition) that are the most adopted in the literature but others could be modelled. In order to add the support for a new task typology in LiquidCrowd, we only need to provide the associated candidate-answer identification technique (i.e., supermajority checks, trustworthiness management, and reward management are defined to comply with any kind of task typology). New task typologies can be defined for example, for long-texts, images, music, and videos.

Application to the social web. A possible application of the presented crowdsourcing approach is to implement a mobile-app for crowd-question&answer. The idea is to build a social-network where users are able to pose and answer questions. The techniques of LiquidCrowd will be used to automatically distinguish among valuable and non-valuable users and to provide a single consensus-based answer to each question. In order to encourage users to answer others questions, we will introduce a penalty-based mechanism reducing worker scores for each posed question.

Incentives mechanisms. In the experimentations presented in Chapter 6, we rely on the incentives called glory and love. Referring to the discussion presented in [Rotman et al., 2012], we plan to introduce an incentive-schema to enforce worker participation by working in two main directions. On one side, we plan to investigate which are the points in which worker participation declines (e.g., at the end of a task, after reaching a certain number of completed tasks). In order to emphasize the workers interest in the research project, the Argo prototype will automatically produce and show to the worker a detailed report about her/his completed tasks. This way, we would better explain the supermajority mechanisms and justify the (eventual) tasks termination, thus enforcing worker participation. On the other side, we plan to highlight data use to increase the workers involvement in the research project. The main idea is to improve the Argo prototype by introducing a new section where workers will be informed about publications/experimentations where the results obtained through their work have been presented/used.

Stereomood experimentation. This experimentation has already been executed and

it is based on emotional recognition of music tracks. In this experimentation, we provided both choice and proposition tasks where workers were asked to choose/propose an emotional label after listening a music track extracted from a repository provided by the web service *Stereomood*[1]. This experimentation had a duration of 30 days and involved 223 workers labelling 267 music tracks with a group size $s_{\mathcal{G}} = 6$. The collected results will be compared with the expert-validated ground-truth provided by Stereomood. Some preliminary results of this comparison show that more than the 50% of the committed choice-tasks produced the same labelling produced by Stereomood experts.

[1] http://www.stereomood.com

Bibliography

Enrique Estellés Arolas and Fernando González-Ladrón de Guevara. Towards an Integrated Crowdsourcing Definition. *Journal of Information Science*, 38(2):189–200, 2012.

John Arthur and Shiva Azadegan. Spring Framework for Rapid Open Source J2EE Web Application Development: A Case Study. In *Proc. of the 6th Int. Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, pages 90–95, Towson, Maryland, USA, 2005.

Daniel W. Barowy, Charlie Curtsinger, Emery D. Berger, and Andrew McGregor. AutoMan: A Platform For Integrating Human-Based and Digital Computation. In *Proc. of the 27th Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA 2012)*, pages 639–654, Tucson, AZ, USA, 2012.

Carlo Batini, Stefano Ceri, and Shamkant B. Navathe. *Conceptual Database Design: An Entity-Relationship Approach*. Benjamin/Cummings, 1992.

Christian Bauer and Gavin King. *Java Persistance with Hibernate*. Dreamtech Press, 2006.

Michael S. Bernstein, Greg Little, Robert C. Miller, Björn Hartmann, Mark S. Ackerman, David R. Karger, David Crowell, and Katrina Panovich. Soylent: a Word Processor With a Crowd Inside. In *Proc. of the 23rd Annual Symposium on User Interface Software and Technology (UIST 2010)*, pages 313–322, New York, NY, USA, 2010.

Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked Data - The Story So Far. *Int. Journal on Semantic Web and Information Systems*, 5(3):1–22, 2009.

Alessandro Bozzon, Marco Brambilla, Stefano Ceri, and Piero Fraternali. Liquid Query : Multi-Domain Exploratory Search on the Web. In *Proc. of the 19th Int. World Wide Web Conference (WWW 2010)*, pages 161–170, Raleigh, North Carolina, USA, 2010.

Alessandro Bozzon, Marco Brambilla, Stefano Ceri, and Andrea Mauri. Reactive Crowdsourcing. In *Proc. of the 22nd Int. World Wide Web Conference (WWW 2013)*, pages 153–164, Rio de Janeiro, Brazil, 2013.

Daren C Brabham. Moving the crowd at threadless: Motivations for participation in a crowdsourcing application. *Information, Communication & Society*, 13(8):1122–1145, 2010.

Anthony Brew, Derek Greene, and Padraig Cunningham. Using Crowdsourcing and Active Learning to Track Sentiment in Online Media. In *Proc. of the 19th European Conference on Artificial Intelligence (ECAI 2010)*, pages 145–150, Lisbon, Portugal, 2010.

Rune Brincker, Lingmi Zhang, and P. Andersen. Modal Identification from Ambient Responses Using Frequency Domain Decomposition. In *Proc. of the 18th Int. Modal Analysis Conference (IMAC 2000)*, San Antonio, Texas, 2000.

Chris Buckley, Matthew Lease, and Mark D. Smucker. Overview of the TREC 2010 Relevance Feedback Track (Notebook). In *Proc. of the 19th Text Retrieval Conference (TREC 2010)*, Gaithersburg, Maryland, USA, 2010.

Simon Butler, Michel Wermelinger, Yijun Yu, and Helen Sharp. Mining Java Class Naming Conventions. In *Proc. of the 27th Int. Conference on Software Maintenance (ICSM 2011)*, Williamsburg, VA, USA, 2011.

S. Castano, A. Ferrara, and S. Montanelli. Clouding Services for Linked Data Exploration. In *Proc. of the Int. Conference on Advanced Information Systems Engineering (CAiSE 2012)*, pages 486–501, Gdansk, Poland, 2012a.

Silvana Castano, Alfio Ferrara, and Stefano Montanelli. Structured Data Clouding Across Multiple Webs. *Information Systems*, 37(4):352–371, 2012b.

Silvana Castano, Alfio Ferrara, and Stefano Montanelli. *Search Computing - Broadening Web Search*, chapter Thematic Clustering and Exploration of Linked Data. Springer, 2013a.

Silvana Castano, Alfio Ferrara, Stefano Montanelli, and Gaia Varese. *Ontology and Instance Matching*, pages 167–195. Springer-Verlag, Heidelberg, Germany, 2011.

Silvana Castano, Lorenzo Genta, and Stefano Montanelli. Leveraging Crowdsourced Knowledge for Web Data Clouds Empowerment. In *Proc. of the 7th IEEE Int. Conference on Research Challenges in Information Science (RCIS 2013)*, pages 1–10, Paris, France, 2013b.

Silvana Castano, Lorenzo Genta, and Stefano Montanelli. Combining Crowd Consensus and User Trustworthiness for Managing Collaborative Tasks. *Future Generation Computer Systems (submitted for publication)*, 2014.

Alexander Philip Dawid and Allan M Skene. Maximum Likelihood Estimation of Observer Error-Rates Using the EM Algorithm. *Applied statistics*, 28(1):20–28, 1979.

Gianluca Demartini, Djellel Eddine Difallah, and Philippe Cudré-Mauroux. ZenCrowd: Leveraging Probabilistic Reasoning and Crowdsourcing Techniques for Large-Scale Entity Linking. In *Proc. of the 21st Int. World Wide Web Conference (WWW 2012)*, pages 469–478, Lyon, France, 2012.

AnHai Doan, Raghu Ramakrishnan, and Alon Y. Halevy. Crowdsourcing Systems on the World-Wide Web. *Communications of the ACM*, 54(4):86–96, 2011.

Steven Dow, Anand Pramod Kulkarni, Brie Bunge, Truc Nguyen, Scott R. Klemmer, and Björn Hartmann. Shepherding the crowd: managing and providing feedback to crowd workers. In *In Proc. of the Int. Conference on Human Factors in Computing Systems (CHI 2011)*, pages 1669–1674, Vancouver, BC, Canada, 2011.

Julie S. Downs, Mandy B. Holbrook, Steve Sheng, and Lorrie Faith Cranor. Are Your Participants Gaming the System?: Screening Mechanical Turk Workers. In *Proc. of the 28th Int. Conference on Human Factors in Computing Systems (CHI 2010)*, pages 2399–2402, Atlanta, Georgia, USA, 2010.

Jianhong Feng, Guoliang Li, Henan Wang, and Jianhua Feng. Incremental Quality Inference in Crowdsourcing. In *Proc. of the 19th Int. Database Systems for Advanced Applications Conference (DASFAA 2014)*, pages 453–467, Bali, Indonesia, 2014.

Alfio Ferrara, Lorenzo Genta, and Stefano Montanelli. Linked data classification: a feature-based approach. In *Proc. of the 3rd EDBT Int. Workshop on Linked Web Data Management (LWDM 2013)*, pages 75–82, Genoa, Italy, 2013.

Alfio Ferrara, Lorenzo Genta, and Stefano Montanelli. Similarity Recognition in the Web of Data. In *Proc. of the 4rd Int. Workshop on Linked Web Data Management (LWDM 2014)*, volume 1133, pages 263–268, Athens, Greece, 2014a.

Alfio Ferrara, Lorenzo Genta, Stefano Montanelli, and Silvana Castano. Dimensional Clustering of Linked Data: Techniques and Applications. *Transactions on Large-Scale Data and Knowledge Centered Systems (accepted for publication)*, 2014b.

Tim Finin, Will Murnane, Anand Karandikar, Nicholas Keller, Justin Martineau, and Mark Dredze. Annotating Named Entities in Twitter Data with Crowdsourcing. In *Proc. of the Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 80–88, Los Angeles, CA, USA, 2010a.

Tim Finin, Will Murnane, Anand Karandikar, Nicholas Keller, Justin Martineau, and Mark Dredze. Annotating Named Entities in Twitter data with Crowdsourcing. In *Proc. of the Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk (NAACL HLT 2010)*, pages 80–88, Los Angeles, CA, USA, 2010b.

Michael J. Franklin, Donald Kossmann, Tim Kraska, Sukriti Ramesh, and Reynold Xin. CrowdDB: Answering Queries with Crowdsourcing. In *Proc. of the Int. Conference on Management of Data (SIGMOD 2011)*, pages 61–72, Athens, Greece, 2011.

David Geiger, Michael Rosemann, and Erwin Fielt. Crowdsourcing Information Systems - A Systems Theory Perspective. In *Proc. of the 22nd Australasian Conference on Information Systems (ACIS 2011)*, page 33, Sydney, Australia, 2011.

Ryan Gomes, Peter Welinder, Andreas Krause, and Pietro Perona. Crowdclustering. In *Proc. of the 25th Annual Conference on Neural Information Processing Systems*, pages 558–566, Granada, Spain, 2011.

Alon Y. Halevy, Anand Rajaraman, and Joann J. Ordille. Data Integration: The Teenage Years. In *Proc. of the 32nd Int. Conference on Very Large Data Bases (VLDB 2006)*, pages 9–16, Seoul, Korea, 2006.

Derek L. Hansen, Patrick John Schone, Douglas Corey, Matthew Reid, and Jake Gehring. Quality Control Mechanisms for Crowdsourcing: Peer Review, Arbitration, & Expertise at Familysearch Indexing. In *Proc. of the 16th Conference on Computer Supported Cooperative Work (CSCW 2013)*, pages 649–660, San Antonio, TX, USA, 2013.

Daniel M. Herzig and Thanh Tran. Heterogeneous Web Data Search using Relevance-based on the Fly Data Integration. In *Proc. of the 21st Int. World Wide Web Conference (WWW 2012)*, pages 141–150, Lyon, France, 2012.

Victoria J. Hodge and Jim Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22(2):85–126, 2004.

Jeff Howe. The rise of crowdsourcing. *Wired magazine*, 14(6):1–4, 2006.

Panagiotis G Ipeirotis, Foster Provost, and Jing Wang. Quality Management on Amazon Mechanical Turk. In *Proc. of the 16th Workshop on Human Computation (SIGKDD 2010)*, pages 64–67, Washington, DC, USA, 2010.

Manas Joglekar, Hector Garcia-Molina, and Aditya G. Parameswaran. Evaluating the Crowd With Confidence. In *Proc. of the 19th Int. Conference on Knowledge Discovery and Data Mining (KDD 2013)*, pages 686–694, Chicago, IL, USA, 2013.

Hyun Joon Jung and Matthew Lease. Improving Consensus Accuracy via Z-Score and Weighted Voting. In *Proc. of the 3rd Human Computation Workshop (HCOMP 2011)*, pages 88–90, San Francisco, California, USA, 2011.

Ece Kamar, Severin Hacker, and Eric Horvitz. Combining Human and Machine Intelligence in Large-Scale Crowdsourcing. In *Proc. of the 11th Int. Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, pages 467–474, Valencia, Spain, 2012.

Aniket Kittur, Ed H. Chi, and Bongwon Suh. Crowdsourcing User Studies with Mechanical Turk. In *Proc. of the Conference on Human Factors in Computing Systems (CHI 2008)*, pages 453–456, Florence, Italy, 2008.

Jon A Krosnick and Leandre R Fabrigar. *Survey Measurement and Process Quality*, chapter Designing Rating Scales for Effective Measurement in Surveys, pages 141–164. John Wiley & Sons, Inc, 1997.

Abhimanu Kumar and Matthew Lease. Modeling Annotator Accuracies for Supervised Learning. In *Proc. of the 4th Int. Workshop on Crowdsourcing for Search and Data Mining (CSDM 2011)*, pages 19–22, 2011.

Kazuhiro Kuwabara and Naoki Ohta. Toward a Crowdsourcing Platform for Knowledge Base Construction. In *Proc. of the 6th Int. Conference on Information, Process, and Knowledge Management (eKNOW 2014)*, pages 89–92, Barcelona, Spain, 2014.

Karim R Lakhani and Zahra Kanji. Threadless: The business of community. *Harvard Business School Multimedia/Video Case*, pages 608–707, 2008.

John Le, Andy Edmonds, Vaughn Hester, and Lukas Biewald. Ensuring Quality in Crowdsourced Search Relevance Evaluation: The effects of training question distribution. In *In Proc. of the 1st workshop on Crowdsourcing for Search Evaluation (SIGIR 2010)*, pages 21–26, Geneva, Switzerland, 2010.

Thomas W Malone, Robert Laubacher, and Chrysanthos Dellarocas. The Collective Intelligence Genome. *IEEE Engineering Management Review*, 38(3):38–52, 2010.

Gary Marchionini. From Finding to Understanding. *Communications of the ACM*, 49 (4):41–46, 2006.

Afra J Mashhadi and Licia Capra. Quality Control for Real-time Ubiquitous Crowdsourcing. In *Proc. of the 2nd Int. Workshop on Ubiquitous Crowdsourcing*, pages 5–8, Bejing, China, 2011.

Winter Mason and Duncan J. Watts. Financial Incentives and the Performance of Crowds. *SIGKDD Explorations Newsletter*, 11(2):100–108, 2010.

Toshiko Matsui, Yukino Baba, Toshihiro Kamishima, and Hisashi Kashima. Crowdordering. In *Proc. of the 18th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining (PAKDD 2014)*, pages 336–347, Tainan, Taiwan, 2014.

Robert McCann, Warren Shen, and AnHai Doan. Matching Schemas in Online Communities: A Web 2.0 Approach. In *Proc. of the 24th Int. Conference on Data Engineering (ICDE 2008)*, pages 110–119, Cancún, México, 2008.

Patrick Minder and Abraham Bernstein. CrowdLang: A Programming Language for the Systematic Exploration of Human Computation Systems. In *Proc. of the 4th Int. Conference on Social Informatics*, pages 124–137, Lausanne, Switzerland, 2012.

Stefano Montanelli, Silvana Castano, and Lorenzo Genta. Urban Information Integration through Smart City Views. *Int. Journal of Knowledge and Learning (accepted for publication)*, 2014.

Gonzalo Navarro. A Guided Tour to Approximate String Matching. *ACM Computing Surveys*, 33(1):31–88, 2001.

Phu Tran Nguyen, Juho Kim, and Robert C. Miller. Generating Annotations for How-to Videos Using Crowdsourcing. In *Proc. of the Conference on Human Factors in Computing Systems (CHI 2013)*, pages 835–840, Paris, France, 2013.

Jon Noronha, Eric Hysen, Haoqi Zhang, and Krzysztof Z Gajos. Platemate: Crowdsourcing Nutritional Analysis from Food Photographs. In *Proc. of the 24th symposium on User Interface Software and Technology*, pages 1–12, Santa Barbara, CA, USA, 2011.

Stefanie Nowak and Stefan M. Rüger. How Reliable are Annotations via Crowdsourcing: a Study about Inter-annotator Agreement for Multi-label Image Annotation. In *Proc. of the 11th Int. Conference on Multimedia Information Retrieval (MIR 2010)*, pages 557–566, Philadelphia, Pennsylvania, USA, 2010.

David Oleson, Alexander Sorokin, Greg P. Laughlin, Vaughn Hester, John Le, and Lukas Biewald. Programmatic Gold: Targeted and Scalable Quality Assurance in Crowdsourcing. In *Proc. of the 1st workshop on Human Computation*, pages 11–11, San Francisco, California, USA, 2011.

Aditya G. Parameswaran, Stephen Boyd, Hector Garcia-Molina, Ashish Gupta, Neoklis Polyzotis, and Jennifer Widom. Optimal Crowd-Powered Rating and Filtering Algorithms. In *Proc. of the 40th Int. Conference on Very Large Data Bases (VLDB 2014)*, volume 7, pages 685–696, Hangzhou, China, 2014.

Aditya G. Parameswaran, Hector Garcia-Molina, Hyunjung Park, Neoklis Polyzotis, Aditya Ramesh, and Jennifer Widom. CrowdScreen: Algorithms for Filtering Data with Humans. In *Proc. of the Int. Conference on Management of Data (SIGMOD 2012)*, pages 361–372, Scottsdale, AZ, USA, 2012.

Vikas C. Raykar, Shipeng Yu, Linda H. Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. Learning From Crowds. *Journal of Machine Learning Research*, 11:1297–1322, 2010.

Dana Rotman, Jenny Preece, Jennifer Hammock, Kezee Procita, Derek L. Hansen, Cynthia Sims Parr, Darcy Lewis, and David W. Jacobs. Dynamic Changes in Motivation in Collaborative Citizen-Science Projects. In *Proc. of the 2012 Int. Conference on Computer Supported Cooperative Work*, pages 217–226, Seattle, WA, USA, 2012.

Senjuti Basu Roy, Ioanna Lykourentzou, Saravanan Thirumuruganathan, Sihem Amer-Yahia, and Gautam Das. Crowds, not Drones: Modeling Human Factors in Interactive Crowdsourcing. In *Proc. of the 1st VLDB Workshop on Databases and Crowdsourcing (DBCrowd 2013)*, pages 39–42, Riva del Garda, Trento, Italy, 2013.

Cristina Sarasua, Elena Simperl, and Natalya Fridman Noy. CrowdMap: Crowdsourcing Ontology Alignment with Microtasks. In *Proc. of the 11th Int. Semantic Web Conference, (ISWC 2012)*, volume 7649, pages 525–541, Boston, MA, USA, 2012.

Ognjen Scekic, Hong Linh Truong, and Schahram Dustdar. Incentives and rewarding in social computing. *Communications of the ACM*, 56(6):72–82, 2013.

Aashish Sheshadri and Matthew Lease. SQUARE: A Benchmark for Research on Computing Crowd Consensus. In *Proc. of the 1st Conference on Human Computation and Crowdsourcing (HCOMP 2013)*, pages 156–164, Palm Springs, CA, USA, 2013.

Elena Simperl, Barry Norton, and Denny Vrandecic. Crowdsourcing Tasks in Linked Data Management. In *Proc. of the 2nd Int. Workshop on Consuming Linked Data (COLD 2011)*, Bonn, Germany, 2011.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. Cheap and fast - but is it good? evaluating non-expert annotations for natural language tasks. In *Proc.*

of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2008), pages 254–263, Honolulu, Hawaii, USA, 2008.

Kate Starbird, Grace Muzny, and Leysia Palen. Learning from the Crowd: Collaborative Filtering Techniques for Identifying on-the-ground Twitterers during Mass Disruptions. In *Proc. of the 9th Int. Conference on Information Systems for Crisis Response and Management (ISCRAM 12)*, Vancouver, Canada, 2012.

Brian L Sullivan, Christopher L Wood, Marshall J Iliff, Rick E Bonney, Daniel Fink, and Steve Kelling. eBird: A citizen-based bird observation network in the biological sciences. *Biological Conservation*, 142(10):2282–2292, 2009.

James Surowiecki. *The Wisdom of Crowds*. Random House LLC, Munich, Germany, 2005.

Vojtech Svátek, Ondrej Sváb-Zamazal, and Valentina Presutti. Ontology Naming Pattern Sauce for (Human and Computer) Gourmets. In *Proc. of the 1st Workshop on Ontology Patterns (WOP 2009)*, 2009.

Wei Tang and Matthew Lease. Semi-supervised Consensus Labeling for Crowdsourcing. In *In Proc. of the 2nd Int. Workshop on Crowdsourcing for Information Retrieval (CIR 2011)*, pages 36–41, 2011.

Matteo Venanzi, John Guiver, Gabriella Kazai, Pushmeet Kohli, and Milad Shokouhi. Community-Based Bayesian Aggregation Models for Crowdsourcing. In *Proc. of the 23rd Int. World Wide Web Conference (WWW 2014)*, pages 155–164, Seoul, Republic of Korea, 2014.

Peter Welinder, Steve Branson, Serge Belongie, and Pietro Perona. The Multidimensional Wisdom of Crowds. In *Proc. of the 24th Conference on Neural Information Processing Systems (NIPS 2010)*, pages 2424–2432, Vancouver, British Columbia, Canada, 2010.

Steven Euijong Whang, Peter Lofgren, and Hector Garcia-Molina. Question Selection for Crowd Entity Resolution. *Proc. of the Very Large Database endowment (PVLDB)*, 6(6):349–360, 2013.

Jacob Whitehill, Paul Ruvolo, Tingfan Wu, Jacob Bergsma, and Javier R. Movellan. Whose Vote Should Count More: Optimal Integration of Labels from Labelers of Unknown Expertise. In *Proc. of the 23rd Annual Conference on Neural Information Processing Systems (NIPS 2009)*, pages 2035–2043, Vancouver, British Columbia, Canada, 2009.

Rui Xu and Donald C. Wunsch II. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, 2005.

Hui Yang, Anton Mityagin, Krysta Marie Svore, and Sergey Markov. Collecting High Quality Overlapping Labels at Low Cost. In *Proc. of the 33rd Int. Conference on Research and Development in Information Retrieval (SIGIR 2010)*, pages 459–466, Geneva, Switzerland, 2010.

Man-Ching Yuen, Irwin King, and Kwong-Sak Leung. A Survey of Crowdsourcing Systems. In *Proc. of the 3rd Int. SocialCom/PASSAT Conference*, pages 766–773, Boston, MA, USA, 2011.

Omar F Zaidan and Chris Callison-Burch. Crowdsourcing translation: Professional quality from non-professionals. In *Proc. of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 1220–1229, Portland, Oregon, USA, 2011.