# Chapter 9
# Human-in-the-Loop Tasks for Data Management, Citizen Sensing, and Actuation in Smart Environments


**Umair ul Hassan and Edward Curry**

**Keywords** Human-in-the-loop · Crowdsourcing · Citizen sensing · Task routing · Smart environments · Dataspaces · Smart environments

## 9.1 Introduction

Humans are playing critical roles in the management of data at large scales, through activities including schema building, matching data elements, resolving conflicts, and ranking results. The application of human-in-the-loop within intelligent systems in smart environments presents challenges in the areas of programming paradigms, execution methods, and task design. This chapter examines current human-in-the-loop approaches for data management tasks, including data integration, data collection (e.g. citizen sensing), and query refinement. A comparison of approaches (Augmented Algorithms, Declarative Programming, and Stand-alone Platforms) that can enable human tasks within data management is presented. The chapter also covers spatial tasks where users within the smart environment are requested to take physical actions in the environment in the form of citizen actuation.

This chapter discusses the design of the human task service and its use within intelligent applications in the Real-time Linked Dataspace (RLD). The rest of this chapter is organised as follows: the concept of the "Wisdom of the Crowds" and crowdsourcing is explained in Sect. 9.2. Section 9.3 examines the challenges to enable crowdsourcing, including issues with task design, task assignment, and user incentivisation. Section 9.4 introduces existing approaches to utilising humans-in-the-loop with comparisons provided in Sect. 9.5. Section 9.6 discusses the human task service of the RLD with emphasis on service-levels, applications, data processing pipeline, data model, and task routing. The chapter concludes with a summary in Sect. 9.7.



E. Curry, *Real-time Linked Dataspaces*,
https://doi.org/10.1007/978-3-030-29665-0_9

## 9.2 The Wisdom of the Crowds

Crowdsourcing has emerged as a powerful paradigm for solving complex problems at a large scale with the help of a group of people [211–213]. The rapid development of web technologies has made it possible for millions of online users to participate in collective efforts or "crowds" formed in response to open calls. People can contribute by performing tasks such as collecting photos, transcribing audio, classifying images, and classifying items [211]. The notion of "wisdom of crowds" advocates that potentially large groups of non-experts can solve complex problems usually considered to be solvable only by experts.

Crowdsourcing has been applied to develop prediction markets, design innovative solutions, and support knowledge generation, with the help of dedicated collaboration platforms such as Wikipedia or online workers to perform micro-tasks through marketplaces such as Amazon Mechanical Turk (AMT). Crowdsourcing would seem like a natural fit to be applied for supporting collaborative data management at large scales where human intelligence can complement machine computation to achieve higher-quality results and capitalise on the domain knowledge of the crowd. The suitability of crowdsourcing for solving complex problems has been demonstrated through several research and industrial projects [214, 215]. In a similar direction, the database research community has started to develop tools and techniques to ease the development efforts required to allow algorithmic access to crowds [216, 217].

While crowdsourcing focuses on combining the effort of a group of human contributors, *human computation* is a paradigm that focuses on an algorithmic approach towards harnessing human affordances [218, 219]. In practice, human computation can leverage crowdsourcing by asking a large number of people to perform specific computational tasks. Quinn and Bederson [218] define human computation based on two aspects of the problem at hand: first, it must follow the general paradigm of computation so that it can be solved by computers alone someday, and second, a computer controls the computational process. Finally, both of these mechanisms can be used to implement human-in-the-loop approaches where both human and machine intelligence are leveraged together. A common example of a human-in-the-loop system is to create machine learning models where humans are directly involved in training, tuning, and testing data for a machine learning algorithm. Crowd correct inaccuracies in machine predictions, thereby increasing accuracy, which results in a higher quality of results. Another common case is to include humans within the feedback-loop of automated decision-making or control systems system.

To demonstrate the application of human-in-the-loop and crowdsourcing, consider the example of an analyst who wants to prepare a list of craft shows, fairs, and festivals ranked according to variety. Instead of searching on the web and sifting through a plethora of websites, the analyst decides to crowdsource this data collection activity. After defining the required attributes of data, such as name, city, URL, and craft variety, the analyst posts a task on Amazon Mechanical Turk. Once the data

collection tasks are complete, the analyst tries to de-duplicate entries with the help of a state-of-the-art algorithm. The algorithm is supplemented with the de-duplication performed by an expert to achieve high-quality results by enabling human tasks. Within the context of data management, human tasks have been demonstrated in different stages of data processing pipeline including the collection and enrichment of data, mapping and matching between schema and records, and feedback and refinement of the results of data quality algorithms.

Within the physical world, crowdsourcing techniques can be used to go beyond data management tasks to ask users to perform real-world "citizen actuation" tasks where users are requested to perform tasks to make a physical change in the environment [220]. The key to enabling the range of human-in-the-loop tasks is the use of a platform to manage your crowd of users.

### *9.2.1 Crowdsourcing Platform*

A crowdsourcing system, in general, has three types of interacting agents: requesters, workers, and platform. Each of these agents is described as follows:

- **Requesters:** Submit tasks to the platform that need to be performed by the crowd. Apart from humans, the requester can also be another application that needs human services for performing its functionality. Requesters are interested in maximising their utility, which is defined in terms of the quality of task performance and the associated costs. Note that the notion of quality and costs can vary between types of tasks and the application domain.
- **Workers:** Members of the crowd who are willing to perform tasks. Workers can vary in terms of their reliability of performed tasks and the incentive they expect for the work. The worker is interested in maximising their utility, which is defined in terms of the effort they exert and the value they gain from performing tasks.
- **Platform:** Serves as the mediator between requesters and workers; therefore, providing the interaction mechanism between both agents. It defines the mode of exchange for tasks, results, feedback, and incentives. A third-party platform provider is interested in maximising the value gained from the use of the software and its functionality. Furthermore, it is in the interest of platform managers to promote the long-term use of their platform.

Figure 9.1 highlights the sequence of interactions between these agents. (1) The requester submits tasks to the platform, which allows filtering of workers based on their characteristics or categories. (2) The tasks are assigned to the appropriate workers. (3) The workers perform the tasks and submit the responses to the platform. The platform assembles the results of crowdsourcing by aggregating and filtering the responses depending on the application domain. (4) The results are sent back to the requesters and (5) feedback on the performance of the workers is shared with the worker assignment component.

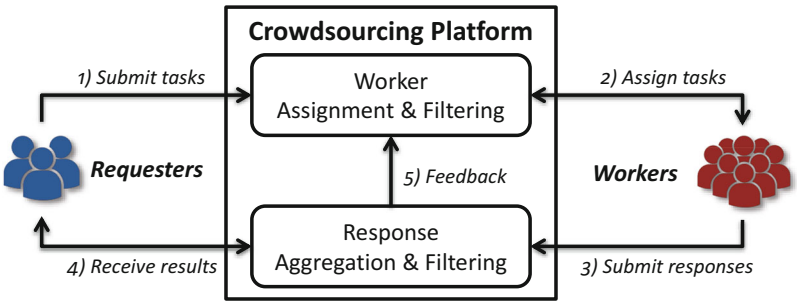


**Fig. 9.1** An overview of a typical interaction between agents in a crowdsourcing scenario [221]

The next section provides a short overview of technical challenges that are required to be addressed for enabling human tasks within smart environments.

## 9.3 Challenges of Enabling Crowdsourcing

The challenges of crowdsourcing entail considerations that are fundamental to enabling contributions from human participants in a smart environment. Law and Ahn [219] provide a detailed overview of human computation in general and its associated challenges, and Li et al. [222] provide a more specific review of the literature on crowdsourcing and data management.

**Task Specification** The first step towards enabling human-in-the-loop is to define the human tasks in terms of input, expected output, and constraints such as conditions for success and time limits. The flexibility of formalisms used for task specification varies among the existing research proposals from declarative methods to algorithmic code [223]. The specification of human tasks also depends on the systems responsible for their execution. For instance, an organisation can deploy a web-based API for allowing different services and applications to access human task services uniformly. Besides the basic specification of a task, additional details can include the details of the composition of a complex task into small tasks and their execution in a workflow [224].

**Interaction Mechanism** Interaction between a human and a human-in-the-loop process requires appropriate user interfaces on a variety of user devices to support the process. It is difficult to design one user interface that fits the requirements of all the various kinds of human tasks [225]. In this regard, existing approaches focus on generating task-specific user interfaces on-the-fly or using templates for different tasks [219]. Besides the design of the element to be presented on a device's screen, there are other factors of interaction mechanism that require careful consideration. For instance, how users will be notified when new tasks need their attention or what interaction mechanisms exist for users looking to perform available tasks. Human

tasks can also be crowdsourced if the number of tasks is sufficiently large, and when the human roles have been defined to perform certain types of tasks. The choice of a crowdsourcing platform requires a trade-off between ease of implementation versus control over worker behaviour. Due to the limited choice of commercial crowdsourcing platforms, most of the existing research prototypes use Amazon Mechanical Turk for crowdsourcing micro-tasks [226]; however, there have been some research projects that also utilised open source platforms such as PyBossa [227].

**Task Assignment** Managing different types of tasks and assigning them to appropriate people is a fundamental challenge of crowdsourcing. The most straightforward approach, as followed by the majority of commercial crowdsourcing platforms, is to allow people to self-assign tasks by searching and browsing through appropriate user interfaces [211]. The alternative approach is to actively make assignment decisions algorithmically, an approach which is better suited for the objectives of human computation [211]. However, the random assignment of tasks to users may not be an effective strategy in domain-specific tasks. For example, domain experts are more suited to accomplish knowledge-intensive tasks that require specific expertise, which may not be available among users of general crowdsourcing platforms [228, 229]. In addition to expertise, the reliability [230] of people in performing tasks, and their physical location [212], can be taken into consideration when matching tasks with workers.

**Result Aggregation** The uncertainty of the correctness of results generated by human computers is an important challenge for enabling human tasks. A well-known solution to this challenge is to employ multiple human computers to perform the same task to ensure the quality of the results. Due to the limited availability of ground truth, the core challenge is to determine whether to accept or reject results generated by a human computer. If we accept results from multiple humans, then the challenge becomes one of how to combine them to generate an aggregated result. Existing approaches to data aggregation from crowdsourced human tasks range from expectation-maximisation algorithms [231] to probabilistic graphical models [232].

**Incentives Mechanisms** Motivating people to participate in human tasks is a challenge for system developers and researchers [233]. Although the accessibility problem has been alleviated due to the plethora of communication tools available for interacting with users over the Internet, the benefit of performing small units of work can be insignificant for the majority of users. The design of appropriate incentive mechanisms is an active area of research in human-in-the-loop and crowdsourcing. While monetary rewards are shown to be most effective in attracting the attention of people [233], other possible solutions include gamification [234] and altruistic motivations [235].

**Latency Issues** Machines and humans differ in the speed of work they can perform in a limited amount of time, which raises the issues of latency in human tasks. The availability of people who have the required knowledge or skill set for a task can also become a bottleneck, requiring changes to computational execution plans [236–

238]. Some efforts have shown that people can be paid to create pools of available crowd workers on-demand [239, 240]; such approaches can be used to support the crowdsourcing of low-latency data [241].

## 9.4 Approaches to Human-in-the-Loop

This section provides an overview of the human-in-the-loop approaches that have been organised into three high-level categories depending on their specialisation of database operations and generality of supporting different human tasks. Some representative examples of existing research are provided for each approach to illustrate the range of applications of human tasks in databases and integration systems.

### *9.4.1 Augmented Algorithms and Operators*

The primary idea of augmented algorithms and operators is to support the data transformation and analytical algorithms with human intelligence. Existing solutions in this category range from database operators (e.g. joins, sorts, skylines) to data integration processes (e.g. schema matching, entity resolution) to data quality updates (e.g. missing values, dependency resolution) [217]. For unstructured data, the algorithmic approaches support natural language processing in activities such as entity extraction and language translation [242]. For multimedia, human computation has been shown to complement a wide range of pattern recognition and machines learning algorithms [231].

Augmented algorithms have been used within dataspaces and databases for feedback-based refinement for data integration. For instance, Roomba is an iterative approach for matching data elements in a dataspace, which solicits human feedback on potential matches [118]. DSToolKit refines schema mappings by asking users to indicate the relevance of query results generated by possible mappings [243]. FICSR generates data constraints from possible schema alignments through an exploration process with the help of experts [244]. Van Keulen and de Keijzer proposed to iteratively reduce the number of possible alternatives in probabilistic data integration processes by soliciting user feedback on ranked query results [245].

The subjective nature of data quality also requires human validation in the data cleaning process. For instance, the GDR system takes a decision-theoretic approach for guiding database repairs through human feedback, and it employs active learning to capture human knowledge for further reduction of user workload [246]. The KATARA system leverages human computation to repair data in a table using a knowledge base [247]. The CrowdAidRepair system combines rules-based and crowd-based data repairing approaches interactively [248].

### 9.4.2 Declarative Programming

Declarative approaches focus on using human computation with the help of well-defined data operations. This approach facilitates independence with respect to the platform used to access human services with the flexibility to access such services within query or programming languages. Extending existing query languages, such as SQL, helps to minimise the learning curve associated with programming human computation. For instance, Deco [223], Qurk [249], and CrowdDB [250] extend SQL to provide database services on top of crowdsourcing platforms. Qurk uses user-defined functions to crowdsource relational comparisons and missing data. Likewise, CrowdDB introduces new SQL keywords to do the same. CrowdDB also allows data collection by defining annotations for columns and tables. Deco focuses on data collection with crowds through the definition of data fetch and resolution rules. Deco separates logical and physical tables to retain crowdsourced raw data. By contrast, hLog is a declarative language to specify human computation during the execution of information extraction and integration programs [251]. hLog extends Datalog with rules to specify details of the required human efforts in information extraction and integration programs. hLog also maintains provenance information to distinguish between crowd-generated versus computer-generated data. The Event Crowd [241] is a hybrid crowd-enabled event processing engine that uses five event crowd operators (Annotate, Rank, Verify, Rate, and Match) that are domain and language independent and can be used by any event processing framework. These operators encapsulate the complexities to deal with crowd workers and allow developers/data scientists to define an event–crowd hybrid workflow.

### 9.4.3 Generalised Stand-alone Platforms

This category of approaches focuses on building platforms with human-in-the-loop functionality, thus providing human intelligence services to other applications in a dataspace. These approaches do not depend on external platforms for human services as compared to previous approaches. For examples, Freebase was a web-based knowledge base that aimed to describe world entities with the help of crowdsourcing. It was supported by a human computation platform called RABj, which allowed users to distribute specific tasks to communities of paid or volunteering humans [252]. RABj provided a programmable interface for access to its human services such as entity matching. The DB-Wiki platform was designed to support collaborative data management with versioning and provenance tracking [253].

## 9.5 Comparison of Existing Approaches

This section compares how existing approaches to human-in-the-loop address the challenges of human tasks and crowdsourcing. Table 9.1 presents a summary. It should be noted that this is a high-level comparative analysis and is not exhaustive in terms of coverage of state-of-the-art literature.

**Task Specification** The complexity of task specification options available increases from augmented algorithms to platforms. Augmented algorithms tend to solve a specific problem within a dataspace with the help of human-in-the-loop techniques; therefore, human tasks are specified using general programming languages. As a result, there is limited agreement on formalisms for task specification among existing research proposals. Declarative approaches use a variety of methods to specify human tasks. The Deco [223] and CrowdDB [250] systems define query languages that extend SQL with new keywords to execute human tasks; additionally, both proposals provide details for query processors that support the execution of new keywords using human processors. By comparison, the Qurk system implemented multiple database operators over a crowdsourcing platform using user-defined functions in SQL. Platforms generally provide RESTful APIs along with their specialised task description models to allow programmable access to human intelligence services [228, 252]; other applications have also developed specialised plugins for existing applications to specify human tasks [251, 254].

**Table 9.1** Specific techniques employed by data management approaches for addressing the primary challenges of human-in-the-loop approaches

| | Augmented algorithms | Declarative programming | Platforms |
|---|---|---|---|
| Task specification | Custom functions | Query language extensions<br>User-defined functions | RESTful APIs<br>Descriptive languages |
| Interaction mechanisms | | Task templates<br>External platform | Task templates<br>Custom UI |
| Task assignment | Online optimisation | | Search & browse UI<br>Task recommendation |
| Result aggregation | Majority votingExpectation maximisation | Majority voting | |
| Incentive mechanisms | Cost Optimisation | Cost Optimisation | Leaderboards<br>Personal Scores<br>Volunteering<br>Auctions<br>Posted Prices |
| Latency issues | Retainer pools<br>Straggler mitigation<br>Pool maintenance<br>Active learning | Tasks batching<br>Response limits<br>Dynamic programming | |

**Interaction Mechanisms** Augmented algorithms either delegate interaction mechanisms, with human computers, to external platforms [248] or design custom user interfaces for each human task [244]. Declarative programming approaches use pre-defined templates to define the user interface of different tasks [249] or automatically generate elements of the user interface from generic templates [250]. Platforms provide task templates, and their associated interfaces [213, 254], or have customised the user interface of existing applications to accommodate human tasks [252].

**Task Assignment** The task assignment strategies for augmented algorithms vary a lot. Some proposals leave the assignment problem to the external platform, while other proposals actively address the assignment problem using online optimisation approaches such as primal-dual and multi-armed bandit models [230]. By comparison, declarative programming approaches for human computation do not directly address the assignment problem; instead, an external platform is used to assign tasks to human computers. Most platforms provide appropriate user interfaces to users for searching and browsing tasks themselves. These user interfaces are complemented with recommendation algorithms based on user profiles [115, 252]. For instance, RABj utilised spatial locality to recommend tasks by matching the current task with the previous history of workers [252], and [255] uses a cost-based algorithm to reduce the travel cost with dynamic task assignment in spatial crowdsourcing.

**Result Aggregation** At the basic level, the use of majority voting for aggregating results of a task from multiple human computers or crowd workers is standard practice [115, 252, 254]. More advanced algorithms employed expectation maximisation based methods to estimate the reliability of workers jointly and generate aggregated results [231]. When dealing with low-quality output, RABj employs voting, and a task escalation strategy that promotes the task to experts for review [252].

Analysis of approaches indicates that the evaluation of the algorithmic approaches is based on incremental improvement over time. On the other hand, declarative and application approaches mostly focus on describing the usefulness of human tasks while describing crowd behaviour. This underlines the lack of standard evaluation methods across approaches, even if they target similar data management problems. Understandably, all of the approaches are based on the relational data model except for Roomba, CAMEE, and RABj, which represent data in graph models [118, 228, 252].

Uncertainty reduction methods are required to improve the quality of outputs generated through human tasks. Active learning uses machine learning in combination with human tasks to generate high-quality results [246]. Providing positive or negative feedback to crowd workers can also improve their future contributions [243, 245]. Provenance information of human-generated data updates can help decide the quality of data [251]. Filtering workers based on their demographical or other characteristics can help reduce spamming [249]. Use of resolution rules for conflicting data updates generated by different workers helps in maintaining high-

quality data [223]. Classifying a worker according to the system's trust in them reduces uncertainty across worker population [254].

**Incentive Mechanisms** The motivation of people to participate in human tasks directly depends on the incentive's mechanism. Both augmented algorithms and declarative programming focus on cost optimisation for human tasks based on rewards. In this regard, the existing approaches have employed a variety of dynamic pricing strategies to minimise costs or control budget [223, 249, 250]. Platforms have employed a wide variety of incentive mechanisms such as leaderboards [252], individual scores [252], posted prices [252, 254], volunteering [228, 252, 254], and auctions [222].

**Latency Issues** Latency issues are addressed through optimisation and heuristics. Latency management approaches for augmented algorithms include creating a retainer pool of pre-fetched workers from external platforms which are shown to have generated data collection results in seconds [239]. The retainer pool approach was further improved using optimisation and active learning [236] and has been applied to work with event systems [241]. Existing proposals for latency issues in declarative programming include batching a group of tasks together, which can reduce the time required to complete individual tasks [249], imposing limits on the time given to crowd workers [249], or employing dynamic programming to minimise latency [237, 238].

## 9.6 Human Task Service for Real-time Linked Dataspaces

Real-time data sources are increasingly forming a significant portion of the data generated in the world. This is in part due to increased adoption of the Internet of Things and the use of sensors for improved data collection and monitoring of smart environments which enhance different aspects of our daily activities in smart buildings, smart energy, smart cities, and others [1]. To support the interconnection of intelligent systems in the data ecosystem that surrounds a smart environment, there is a need to enable the sharing of data among intelligent systems.

### *9.6.1 Real-time Linked Dataspaces*

A data platform can provide a clear framework to support the sharing of data among a group of intelligent systems within a smart environment [1] (see Chap. 2). In this book, we advocate the use of the dataspace paradigm within the design of data platforms to enable data ecosystems for intelligent systems.

A dataspace is an emerging approach to data management that recognises that in large-scale integration scenarios, involving thousands of data sources, it is difficult and expensive to obtain an upfront unifying schema across all sources [2]. Within dataspaces, datasets *co-exist* but are not necessarily fully integrated or homogeneous

in their schematics and semantics. Instead, data is integrated on an *as-needed* basis with the labour-intensive aspects of data integration postponed until they are required. Dataspaces reduce the initial effort required to set up data integration by relying on automatic matching and mapping generation techniques. This results in a loosely integrated set of data sources. When tighter semantic integration is required, it can be achieved in an incremental *pay-as-you-go* fashion by detailed mappings among the required data sources.

We have created the Real-time Linked Dataspace (RLD) (see Chap. 4) as a data platform for intelligent systems within smart environments. The RLD combines the pay-as-you-go paradigm of dataspaces with linked data, knowledge graphs, and real-time stream and event processing capabilities to support a large-scale distributed heterogeneous collection of streams, events, and data sources [4]. Within this section, we focus on the human task support service of the RLD.

### 9.6.2 Human Task Service

The human task service is concerned with providing humans-in-the-loop services to applications within the RLD. The service supports both virtual tasks (data management) and physical tasks (citizen serving) within the smart environment. Collaborative data management [115] within the RLD is enabled by distributing small data management tasks among willing users in the smart environment [221, 256]. The inclusion of users in the data management process not only helps in managing data but may help in building user trust and a sense of ownership of the dataspace.

Figure 9.2 shows a simple architecture for the human task service that includes a *Task Management* component which provides middleware for access to the users in the smart environment. Task management is decoupled from the data management for encapsulation. The core functions of the task management are: (1) *Task Assignment:* matching between tasks and users in the smart environment [256] based on characteristics of tasks or the specific requirements of tasks in terms of human capabilities [115, 221], and (2) *Quality assurance* to ensure truthful and correct responses of tasks.

### 9.6.3 Pay-As-You-Go Service Levels

Dataspace support services follow a tiered approach to data management that reduces the initial cost and barriers to joining the dataspace. When tighter integration into the dataspace is required, it can be achieved incrementally by following the service tiers defined. The incremental nature of the support services is a core enabler of the pay-as-you-go paradigm in dataspaces. The functionality of the human task service follows the 5 Star pay-as-you-go model (detailed in Chap. 4) of the RLD. The human task service has the following levels of incremental support:

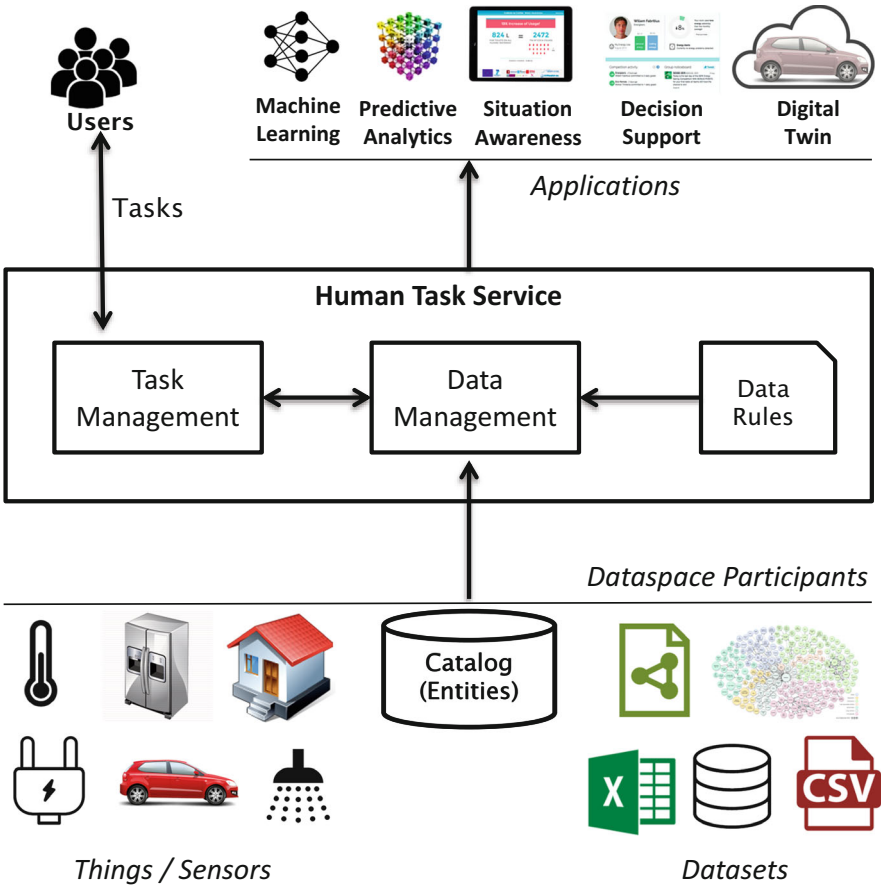


**Fig. 9.2** Overview of the human task service for Real-time Linked Dataspace [4]

*1 Star* **No Service:** No human tasks are used.
*2 Stars* **Schema:** Tasks are used to map schemas among sources.
*3 Stars* **Entity:** Tasks are used to map entities among sources.
*4 Stars* **Enrichment:** Tasks are used to enrich entities with contextual data.
*5 Stars* **Data Quality:** Tasks are used to improve the quality of data (e.g. verification).

### *9.6.4 Applications of Human Task Service*

The Human Task service may be called by applications using the RLD, or by other support services within the RLD support platform. The following four categories of human tasks are fundamental in supporting the data processing pipeline in the RLD.


**Fig. 9.3** Examples of a human task to enrich entities. (**a**) Sensor metadata enrichment. (**b**) Entity enrichment (e.g. room features) [4]

- **Collection and Enrichment:** These types of tasks involve provisioning of data related to entities of interest in a dataspace by humans. A good illustrative example of the service in action is human tasks for entity enrichment [256]. Based on their knowledge and understanding of the environment, users can help to enrich important entities in the dataspace. Figure 9.3a shows an example enrichment task that is associated with an IoT device (e.g. CoAP sensor). The human task can be retrieved by scanning the QR code on the device with a mobile phone or tablet device. The task is retrieved from the human task service, and the user is asked a set of simple questions about the surroundings of the sensor to enrich the description of the entity in a smart building (e.g. Fig. 9.3b: what are the features of the room where the sensor is located?). Similar human tasks for data enrichment can be used across different forms of media in a dataspace, (e.g. image or video annotation).
- **Mapping and Matching:** Finding or verifying mappings among data elements of schemas and entities is another fundamental task of data integration in a dataspace that is suitable for human-in-the-loop and crowdsourcing. For example, crowdsourcing is successful in aligning ontologies in the biomedical domain [257]. A set of generalised solutions for entity resolution and matching have been proposed where they exploit human tasks to generate accurate results [258–261].
- **Operator Evaluation:** Human tasks for supporting the evaluation of database operators allow manipulation and analysis over data-in-motion and data-at-rest. This includes standard databases' operators and queries such as sort, join, and rank. For instance, human-powered evaluation of such database operators has been demonstrated for sorting [249] and skyline queries [262].
- **Feedback and Refinement:** In addition to the above human tasks, a more general set of tasks involve supporting algorithms for improvement of data quality processes, analytical models, and data transformation processes based on

feedback provided on subjective correctness or relevance of results generated by data management algorithms in the dataspace. This can range from verifying data repairs [263] to entity recognition in natural language processing [242], to providing labelled data for machine learning algorithms [264].

- **Citizen Actuation:** Moving to more physical tasks, citizen actuation tasks are formally defined as the activation of a human being as the mechanism by which a control system acts upon the environment [220]. These tasks request users to complete a physical action in the environment, such as closing a window or turning off a light. Citizen actuation tasks can take place in small-scale smart environments, like a neighbourhood/community, to medium and large-scale environments, such as a city. We envisage citizen actuation as forming part of the design process of future smart devices and environments as a method to keep users engaged with their surroundings [265].

### *9.6.5 Data Processing Pipeline*

The general pipeline for data processing in the RLD can be captured in the following four steps:

- **Data Definition:** The first step is to define the semantics and schematics of information to be processed and analysed. At the schema level, this includes the definition of concepts, entities, and their relationships, as well as specific attributes of entities. While basic semantics can be specified in the form of simple vocabularies and constraints, a more detailed semantic representation may require formal ontologies. Within the RLD, the catalog and entity management service (see Chap. 6) is used to maintain entity metadata.
- **Data Collection:** Where data acquisition is needed, the required data is collected from users by manual entry or automated tools. For example, filling out online forms, mobile applications, and entering data to a specific spreadsheet are all methods of manual data collection.
- **Data Integration and Quality:** Given that a dataspace spans multiple datasets and data sources, integration of data is a fundamental task in dataspaces that involves overcoming semantic and schematic heterogeneities of different datasets in order to present a common view of real-world entities. Data integration is performed in conjunction with data quality improvement, which can involve matching and de-duplication of schema elements and individual entities.
- **Data Analysis and Visualisation:** Users and applications pose analytical, and visualisation queries over integrated, high-quality data. Such queries are either used for creating a specific machine learning and statistical model or for serving data through appropriate graphical interfaces on different devices as required for decision-making and analysis.

Humans participate in nearly all stages of this pipeline assuming different roles and expertise such as administrators, data entry operators, integration developers, data analysts/scientists. Data administration includes but is not limited to access control, data serialisation, query management, replication, and fault tolerance. A

considerable amount of research has been dedicated to the automation of data processing pipelines. However, automation solutions suffer from accuracy issues and produce uncertain results, thus requiring constant human supervision.

### 9.6.6 Task Data Model for Micro-tasks and Users

We used the *Semantically Linked Users and Actions* (SLUA) ontology for modelling micro-tasks and users [266] within the human task service. Figure 9.4 outlines the main classes and properties in the SLUA ontology.

The main classes of the ontology are:

- **Action:** Represents a specific act that is performed by the members of the crowd. An action can be cognitive or physical. For example, the comparison of two images involves a cognitive action from the user.
- **Task:** Defines the unit of work resulting in the desired outcome that is assigned to the members of the crowd. A task may require one or more actions to produce the outcome. Therefore, a task at the lowest level is composed of actions. The Task class has a composition relationship with itself because complex tasks can be broken down into small, simple tasks.
- **User:** The class that describes the human contributor in crowdsourcing. The user serves as an intelligent agent that can perform actions for the successful completion of assigned tasks.
- **Reward:** Associated with a task as the incentive for human contribution.
- **Capability:** Defines the human characteristics that are necessary for performing a task. For instance, one system might specify a user's location capability, while another system utilises this description to assign tasks relevant to the same location.

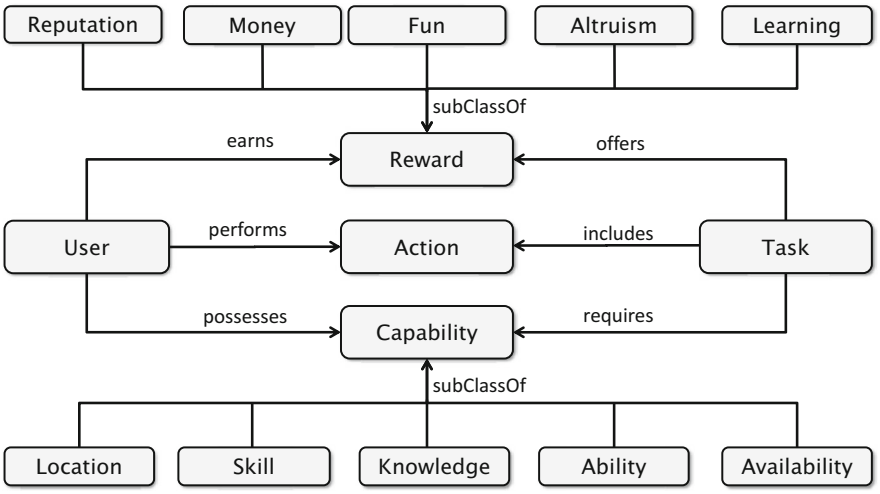


**Fig. 9.4** The SLUA ontology for micro-tasks and users

The main properties of the SLUA ontology used by the human task service are described below:

- **Domain:** A domain definition applies to most of the classes defined above. This property can be helpful for domain-specific algorithms. A common categorisation system could be used to specify domains in general crowdsourcing systems. However, for specific areas, purpose-built taxonomies can be more effective.
- **Offers:** This property defines the relationship of Reward with Task. For example, some tasks might be rewarded with money. By comparison, a user who is interested in a particular reward can be described with the "earns" property.
- **Requires:** A Task can define requirements of one or more human capabilities using this property. In contrast, a User can be described by having similar capabilities using the "possesses" property.
- **Includes:** A Task can define one or more actions that a User performs for generating the desired outcome of a task.
- **isPartOf:** A complex Task can be decomposed into small manageable tasks. Therefore, this property helps in describing the composition relationship between tasks.
- **hasDeadline:** This property can be used to specify the time limitations of a Task, which is specifically important for real-time systems employing crowds.
- **isConnectedWith:** In the context of social networks, users are connected with other users through various relations. This property captures the network structure of users to enable social network based analysis of actions and users. For example, the network structure can be exploited to recommend actions to neighbour nodes in a network.

### *9.6.7 Spatial Task Assignment in Smart Environments*

A major challenge in spatial crowdsourcing is to assign reliable workers to nearby tasks. The goal of such a task assignment process is to maximise the task completion in the face of uncertainty. This process is further complicated when tasks arrivals are dynamic, and worker reliability is unknown. Effective assignment of tasks to appropriate users at the right time is critical to dynamic smart environments. Therefore, information about a user's location and availability are required to assign tasks related to devices and "Things" around them. There are a variety of methods for sourcing the location and availability information of users. In the human task service, we use a mixture of pull and push methods for sourcing a participant's location for making assignment decisions.

**Task Pull** The linkage between physical sensing devices and tasks is made with the help of Quick Response (QR) codes. Figure 9.5 illustrates an example of a QR code attached to a sensor. Each QR code represents an encoded URL for the sensor tasks. Resolving the URL through a browser renders the tasks associated with the sensor. The pull-based assignment is suited for situations where the information about sensors and users is not available.

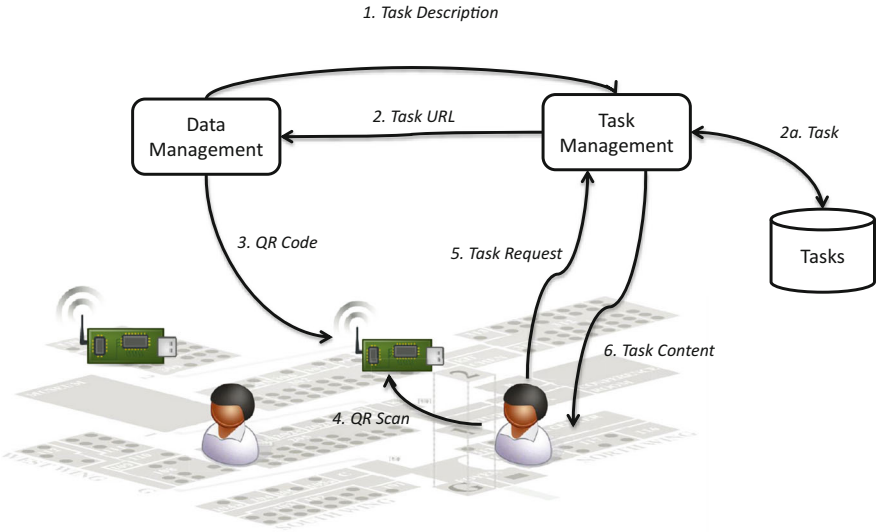


**Fig. 9.5** Example of a pull-based approach to task routing in a smart environment

**Task Push** The tasks can be actively pushed to users in situations where the location and availability information of users is available (see Fig. 9.6). To achieve this, the human task service keeps track of the sensor location information submitted by users and then pushes tasks that require the description of the surroundings of a sensor to nearby users.

Whether the pull or push method is used for assignment depends on the data management objective of the task. Therefore, the data management component must specify the assignment method for tasks, as well as the requirements in terms of human capabilities [213, 266]. Given a set of workers and their associated locations, we employ a graph-based approach to calculate their distance from the task location (see Fig. 9.7). Subsequently, the distance vector is used to optimise the assignment of a task to the appropriate worker. The optimisation objective is maximising the success rate of task completion. For this purpose, we employ an online assignment approach that aims to assign tasks to the best users while also learning about their task acceptance behaviour.

We developed a task assignment algorithm based on the multi-armed bandit model [267] as illustrated in Fig. 9.8. Our task assignment algorithm proceeds in the following manner.

- The algorithm considers the current task and the current pool of workers together with the distance vector. The vector contains the distance variables defined according to the task and worker locations.
- The algorithm chooses a worker and assigns the current task to them, based on the above information and the observed history of the previous assignments.

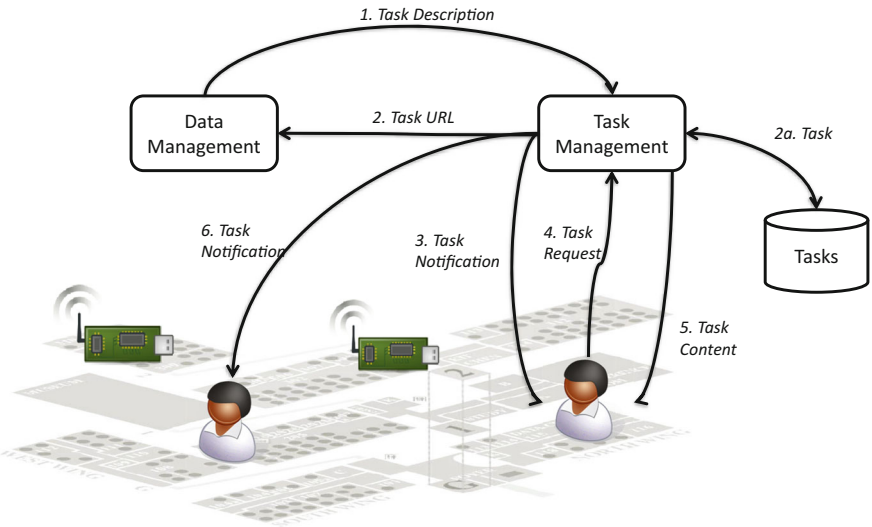


**Fig. 9.6** Example of a push-based approach to task routing in a smart environment

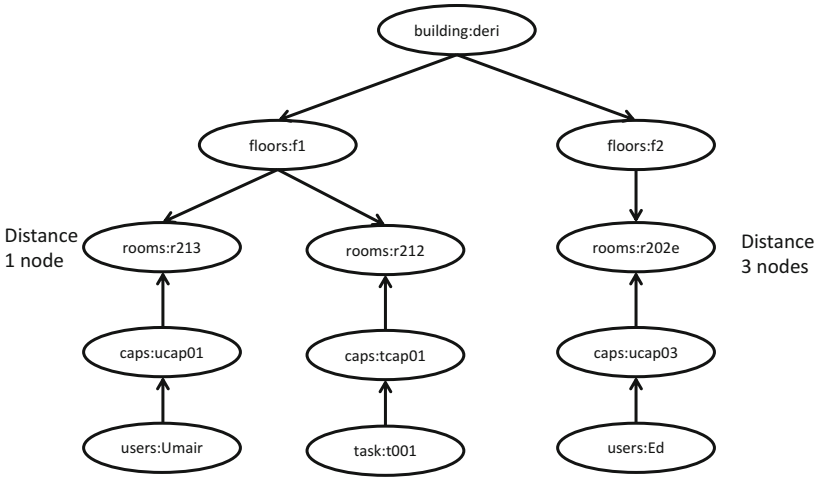


**Fig. 9.7** Hierarchical approach to a capability-based approach to matching the location of task and user

- For each assignment, the algorithm waits for the response, which depends on the current task and the chosen worker. If the worker performs the task before the expiry time, then the completed task counter is updated for the worker.

The algorithm improves its worker selection strategy based on the new observed history of the assigned tasks. The algorithm does not observe any information from the workers that are not chosen for the assignment.

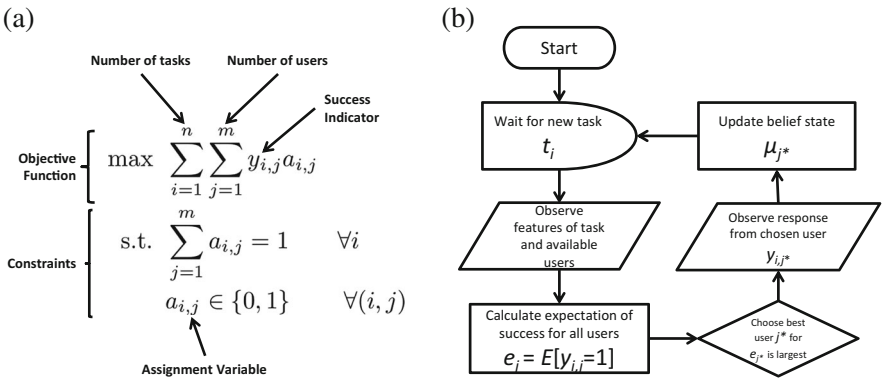


**Fig. 9.8** Assignment algorithm. (**a**) Optimised task assignment using dynamic programming. (**b**) Multi-armed bandit approach for learning

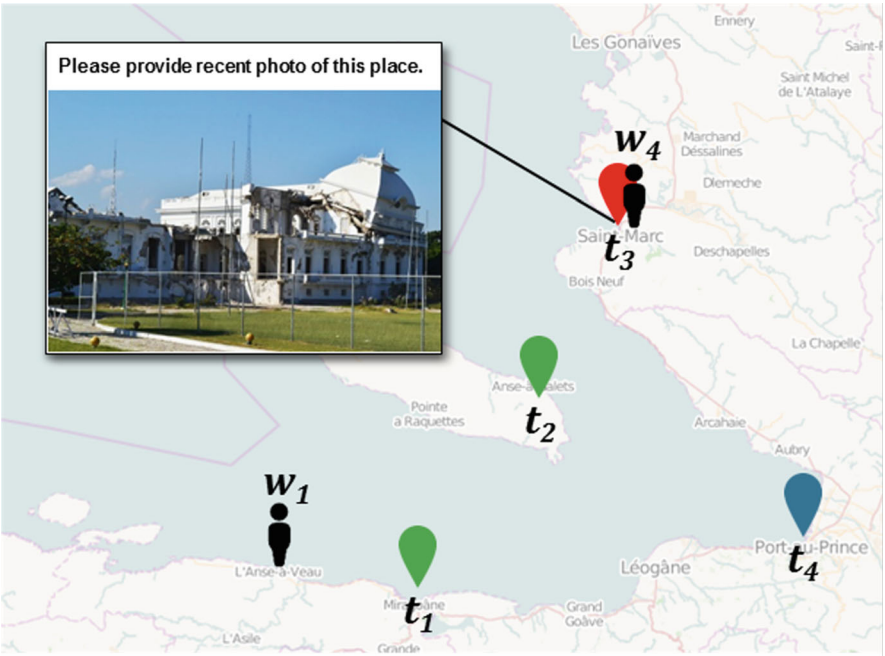


**Fig. 9.9** Example of spatial crowdsourcing on the map of Haiti after the 2010 earthquake. A new spatial task (in blue) requests recent photos of a building at the indicated location [221]

Within larger smart environments, the routing problem becomes more complex, and the assignment algorithms need to be more sophisticated. Consider the situation where a requester is interested in collecting high-quality and representative photos of disaster-hit areas in a country. The locations of interest are spread across the country. The requester designs a task for each location and is interested in the coverage of all locations with high-quality results. Figure 9.9 illustrates such a scenario on a map.

In large-scale deployments, the human task service uses a task assignment approach that combines a bi-objective optimisation with combinatorial multi-armed bandits. We formulate an online optimisation problem to maximise task reliability and minimise travel costs in spatial crowdsourcing. With the use of the *Distance-Reliability Ratio* (DRR) algorithm based on combinatorial fractional programming, we can reduce travel costs by 80% while maximising reliability when compared to existing algorithms [255]. The DRR algorithm has been extended for the scenario where worker reliabilities are unknown by using an *interval estimation* heuristic to approximate worker reliabilities. Experimental results demonstrate that the approach achieves high reliability in the face of uncertainty.

## 9.7 Summary

This chapter provides an introduction to the use of human-in-the-loop and crowdsourcing in intelligent systems within smart environments, where it can be used for virtual tasks (e.g. data management) and physical tasks (e.g. citizen actuation). The use of human-in-the-loop approaches offers exciting opportunities for utilising human processing in making sense of the data deluge and in interacting with the physical environment. However, it requires new ways of thinking about algorithms and platforms while being aware of information security, privacy, and worker exploitation issues. Within the Real-time Linked Dataspace, we have created a Human Task Service as part of the support platform. The purpose of the service is to offer human-in-the-loop support to applications within the dataspace. This chapter presented the design of the human task service and its use within human-in-the-loop applications in a smart environment, its data processing pipeline, data model, and task routing mechanisms.