HEADWORK: Powering the Crowd with Tuple Artifacts
David Gross-Amblard, Marion Tommasi, Iandry Rakotoniaina, Constance Thierry, Rituraj Singh, Léo Jacoboni

To cite this version:
David Gross-Amblard, Marion Tommasi, Iandry Rakotoniaina, Constance Thierry, Rituraj Singh, et al.. HEADWORK: Powering the Crowd with Tuple Artifacts. 2023. hal-04162652

HAL Id: hal-04162652
https://inria.hal.science/hal-04162652
Preprint submitted on 15 Jul 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License
HEADWORK: Powering the Crowd with Tuple Artifacts

David Gross-Amblard
first.last@irisa.fr
Univ Rennes / Irisa Lab
France

Marion Tommasi
Inria Lille-Nord Europe
Université de Lille
France

Constance Thierry
first.last@irisa.fr
Univ Rennes / Irisa Lab
France

Rituraj Singh
Univ Rennes / Irisa Lab
France

Iandry Rakotoniaina
Université de Lille
France

Leo Jacoboni
Wirk
France

ABSTRACT
In this demo we introduce HEADWORK, an open-source academic platform for the crowdsourcing of complex tasks. Besides classical crowdsourcing features, HEADWORK eases the development of crowdsourcing campaigns through a full relational abstraction of relevant concepts (participants, skills, tasks, current answers, decision procedures, GUI, etc.). It allows in particular the orchestration of complex dynamic tasks using so-called tuple artifacts (i.e. finite-state automata which transition guards and actions are SQL-defined, on an evolving database). The demo will illustrate these key features, both from the participant and developer point of view.

CCS CONCEPTS
• Information systems → Database management system engines; • Human-centered computing → Computer supported cooperative work.

KEYWORDS
crowdsourcing, tuple artifacts, business artifacts, macro-tasks

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2023 Association for Computing Machinery.

1 https://www.mturk.com/

1 https://www.mturk.com/

1 https://docs.pybossa.com/

1 https://headwork.irisa.fr

1 https://headwork.irisa.fr

1 INTRODUCTION
Crowdsourcing is now a well-established technique to solve tasks that remain difficult for computers, by automatically asking questions to humans. Successful examples are Zooniverse [14], Foldit[5] for participative science, and Amazon Mechanical Turk for rewarded tasks, to name a few.

At the core of crowdsourcing platforms are micro-tasks: simple questions awaiting for a simple answer. A typical example is to identify the polarity of a tweet (aggressive, friendly), a task still hard for machines.

While the crowdsourcing of micro-tasks is well studied, recent works turn their attention to macro-tasks [9], that require a chain of interactions with humans, using various steps and intermediate decisions. A natural application is the crowdsourcing of report writing, where several participants with complementary skills work on different parts of the report, vote for modifications, check contributions, add pictures, etc. Several systems has been considered to handle this kind of tasks [1, 11], but they rely on a low-level, procedural description of interactions. For the task designer, this requires to take care of technical aspects such as graphical user interface, task synchronization, participant interactions, spammer detection, gold answers, answer aggregation, or participant selection methods.

In this demo, we propose to leverage on these previous efforts. We present HEADWORK, a ready-to-use, academic crowdsourcing platform for the deployment of complex tasks. In order to limit the task designer’s efforts, the HEADWORK platform proposes a full relational abstraction of relevant crowd concepts (participants, skills, tasks, GUI,...) and algorithms (task assignments optimization, crowd decision primitives such as majority voting or expectation maximization). The orchestration of macro-tasks is realized through tuple artifacts [6], that are finite state automata operating on a database, which transitions are guarded by SQL conditions and which trigger SQL actions.

To promote the adoption of HEADWORK, the platform is fully open-source (AGPL), and a demo server is available. On the developer’s side, participant interactions can be customized through HTML/Javascript templates. The platform has already been used for participative science campaigns, and is compatible with rewarded crowdsourcing.

In the sequel we position our work with respect to the state of the art, then introduce the model HEADWORK relies on. After presenting the overall platform architecture, we will describe our demo scenarios and conclude with perspectives.

2 RELATED WORK
With the expansion of participatory work, many crowdsourcing platforms have been developed by industries, such as Amazon Mechanical Turk. However, industrial platforms do not always meet the needs of the academic world and new academic platforms have
started to emerge. These platforms are mainly used for the composition or annotation of corpora. We can for example mention Galaxy Zoo\(^4\) [7], a platform where the contributor annotates photos of galaxies according to their shape. There are other accomplished academic platforms, but they deal with very specific themes. For example Siminchikkunarayku\(^5\) was developed [13] to collect data for the preservation of the Peruvian native language. Siminchikkunarayku has the advantage of being open-source, which is the case of few platforms. The gMission [4] system is another open-source crowdsourcing platform for task completion in a specific geographical space. The system makes micro-task recommendations based on the geolocation of contributors.

Most of the systems in the literature have not resulted in platforms, and when they do, the platform focuses on a very specific topic. On the generic side, the major participative science platform is Zooniverse [14]. It allows to design workflows of tasks ranging from text forms and image annotations. Up to our knowledge, accepted workflows are linear deterministic ones (as in a survey) and participant skills are not taken into account. A procedural control of the workflow is technically possible through the Caesar extension.

The idea to propose a relational abstraction for crowdsourcing has been proposed in earlier works, ranging from SQL [3, 10] to Datalog [8]. But their focus is on micro-tasks only, with no specific participant modeling or extension tools.

In the next section we present our model to deal with macro-tasks in a data-oriented style.

3 MODEL

The Headwork platform is data-oriented. Our goal is to focus on transforming data from the crowd rather than dealing with low-level programming issues. We illustrate below our relational abstraction, the template mechanism, and explain the deployment of micro and macro-tasks.

3.1 Relational Abstraction

Several built-in tables are available. Basically:

- The user table gathers information about crowd participants;
- The skill table contains skill definitions (as keywords and levels of expertise), used for tasks and user profiles;
- The template table provides classical user interactions (expressed in HTML and Javascript);
- The task table contains the questions for the crowd;
- The profile table allows to specify which skill is relevant for a task;
- the answer table saves participant contributions and intermediate computations.

Micro-tasks are then built on these notions.

3.2 Micro-Tasks

Headwork comes with different language flavour. A domain-specific language that we call Crowdy is available, allowing to express simply a wide variety of micro-tasks. For example the following code (Listing 1) will propose the question 'Please count the number of snow leopards in the following image' to any participant, favoring those having the wildlife skill. An integer answer is required, and the corresponding error message is provided (HTML and SQL details are omitted for clarity).

```
Listing 1: Leopard counting micro-task (Crowdy)
prepare task 1 as integer input
pick at random IMG from ImageTable(url)
use 'Please_count_the_following_image_IMG' as body
skill 'wildlife' is relevant for the task
launch task
```

The Crowdy language is translated into SQL expressions on our model (Listing 2).

```
Listing 2: Leopard counting micro-task (sugar-free code)
@IMG:= select url from ImageTable
order by (rand(hash(url))) limit 1;
@BODY:= 'Please_count_the_following_image:@IMG';
@CHECKER:= int;
@CHECKERMSG:= 'Please_enter_an_integer';
insert into task(id,body,checker,checkermsg)
values (1, @BODY, @CHECKER, @CHECKERMSG)
insert into profile(1, 'wildlife');
```

If needed, task designers have full control of the SQL counterpart. SQL expressions can also be used in specific Crowdy statements. For example the following lines will pick a question according to its current priority in the database.

```
use (select text from questionList
order by priority desc limit 1)
as body
```

It is noteworthy that letting the task designer access to a full SQL engine is a potential security threat. We will come back to this question in the next section.

3.3 Template Mechanism

Headwork comes with an extensible template mechanism, that allows the task designer to re-use typical crowd interactions, but also to propose new ones to the community. Basic templates are classical HTML form-like inputs such as text, text area, lists and radio buttons. More sophisticated templates are selectors for geographical maps (point of interest, area of interest), image selectors. Audio/video playing (for speech-to-text translation) and audio recording (for text-to-speech) are also available.

The general architecture of a template is an HTML snippet whose interaction is driven by a Javascript code. The code can contain text tags that are populated by a Crowdy statement (as we did above with the IMG tag). The only constraint is to provide the output as a specific field in JSON format, so that Headwork is able to process it into the answer table (note that for security reasons, a

\(^4\)https://www.zooniverse.org/projects/zookeeper/galaxy-zoo/

\(^5\)https://www.siminchikkunarayku.pe/
new template has to be inspected by the platform manager before inclusion, as in any application store).

### 3.4 Macro-Tasks

In short, macro-tasks are workflows of simple tasks, which order and content can evolve according to participant answers and crowd decisions. In Headwork, a macro-task is driven by a *tuple artifact* (Figure 1): a finite state automaton which transition conditions (guards) and actions are expressed in Crowdy (hence SQL at a low level).

Generally speaking, a transition in a tuple artifact has the following structure:

\[
\text{state } s \xrightarrow{\gamma} \text{ state } s', \quad \text{actions: } \alpha
\]

meaning that, if we are in state \( s \) with database \( DB \), and the guard query \( \gamma(DB) \) is true, then we go to state \( s' \), with the new database \( \alpha(DB) \).

The following simple example organizes the counting of snow leopards (Listing 3). We start (launch state) by launching the previous, Listing 1 micro-task (actions) and then jump to the count state (no guard). When 10 answers have been given (guard to reach the decide state), we conclude by choosing a count of snow leopards. We use weighted majority voting, where participants with the relevant skills (here wildlife) have more influence on the final decision. The result part is a view defining the result of the crowd campaign.

![Figure 1: A tuple-artifact for snow leopard counting](image)

**Listing 3: Macro-task description**

```plaintext
launch \rightarrow count
  guard: none
  actions:
    <code from Listing 1>

count \rightarrow decision
  guard: task 1 has 10 answers
  actions:
    take skill-weighted majority(answer)

decide: final

result: last answer
```

Computations and aggregations benefit from the full power of SQL, extended with crowd-style operators such as majority voting. Specific cohort of participants can be defined thanks to queries on the skill and profile tables.

The tuple artifact in Figure 3 depicts a more sophisticated macro-task for which a crowd of 100 respondents is gathered. Participants are then asked to spot leopards or rocks in a collection of images. When at least 3 spots given by different participants match, a relevant element is considered to be identified. Then, depending on the element type (leopard, rock), a corresponding expert is questioned. A consensus of two experts is required to make a decision.

### 4 THE HEADWORK PLATFORM

The platform is organized as follows (Figure 4). Task providers submit a job as a JSON file encoding the crowd data oriented workflow, in the SQL or Crowdy language, based on the various available templates (HTML forms, Maps, Sound I/O, custom Javascript, ...). The workflow engine (written in PHP hosted by Apache) then processes the automaton and render tasks to participants through the Web interface (Javascript, Bootstrap, Figure 2). Participants can create an account, give their profile (skills), see the list of available tasks ranked according to their skills, and start contributing. All information are available in a Mysql database. If required, Headwork is compatible with a rewarded pool of participants through the Wirk service, to speed-up macro-tasks that could not wait for benevolent participants.
For the demonstration, we will start by a basic crowdsourcing interaction for image annotation, where participants are invited to create an account, give some skills and annotate a wildlife image, and see how decisions are made using majority voting (this introduction can be skipped). Then we will demonstrate the flexibility of the interface with controlled text, HTML forms, maps and audio. We will illustrate how the chaining of questions, and the content of questions can be based on the participant previous answers or by crowd decisions, which already goes beyond the capabilities of popular form engines such as Google Form or LimeSurvey. We will show how complex computations can be made using all the power of SQL extensions such as geographical primitives on maps. A preview of the platform is available at https://headwork.irisa.fr.

6 CONCLUSION AND FUTURE WORK

In this demo, we presented Headwork, an open-source crowdsourcing platform. While most platforms are limited to micro-tasks, Headwork allows the monitoring of complex dynamic macro-tasks through tuple artifacts. To do so, the essential concepts of crowdsourcing (participants, skills, tasks...) are abstracted in a relational way.

Our hope is to make Headwork an academic laboratory for studies in macro-task crowdsourcing, while hosting real participative and citizen science projects. In the short future we plan to implement richer, hierarchical skill models [12] and to allow for automatic workflow verification [2].

ACKNOWLEDGEMENTS

We would like to thank the numerous interns from Rennes University that contributed to pieces of this project. This work was also partially funded by the French National Research Agency (ANR) grant Headwork\(^6\) (ANR-16-CE23-0015).

\(^6\)https://headwork.irisa.fr/headwork-web/
REFERENCES


