HEADWORK: a Data-centric Crowdsourcing Platform for Complex Tasks and Participants

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

Marion Tommasi
Inria Lille-Nord Europe
Université de Lille
France

Iandry Rakotoniaina
UMR 7204, MNHN-CNRS-UPMC,
CESCO, Paris
France

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

Rituraj Singh
Univ Rennes / Irisa Lab
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 illustrates these key features, both from the participant and developer point of view.

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[1] 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 [2] [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[3] (AGPL), and a demo server is available[4]. 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*[5] [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 and only few are open source, among them *Siminchikkunarayku*[6] developed [13] to collect data for the preservation of the Peruvian mother tongue, or *gMission* [4] a crowdsourcing platform for task completion in a specific geographical space. The system recommends micro-tasks 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 to image annotations. Up to our knowledge, accepted workflows are linear deterministic ones (as in a survey) and participant skills are not taken into account. Yet, a procedural control of the workflow is technically possible through the Caesar extension. It is noteworthy that a wide range

[1] https://www.mturk.com/
[2] https://docs.pybossa.com/
[3] https://gitlab.inria.fr/druid-public/headwork
[4] https://headwork.irisa.fr
[5] https://www.zooniverse.org/projects/zookeeper/galaxy-zoo/
[6] https://www.siminchikkunarayku.pe/

of worflow (e.g. BPMN) management systems exist[7], but they focus on interactions with a single customer or a single team, without considering a whole crowd.

The idea to devise 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 ... 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 ... image @IMG';
@CHECKER:= int;
@CHECKERMSG:= 'Please enter an integer';
insert into task(id,formbody,checker,checkermsg)
values
    (1,@BODY,@CHECKER,@CHERCKERMSG);
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
```

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 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 new template has to be inspected by the platform manager before inclusion, as in any application store). The `answer` table can contain plain JSON data, that can be extracted in SQL with XPath-like expressions (thanks to MySQL extensions). For complex JSON, a relational mapping can be given to extract data properly. This allows to handle complex/hierarchical annotation tasks that are common e.g. in NLP.

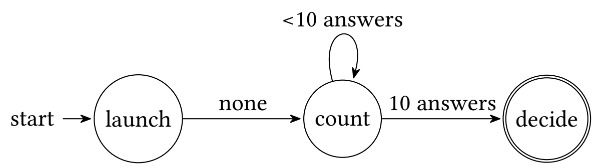

Figure 1: A tuple-artifact for snow leopard counting

3.4 Macro-Tasks

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). The motivation for using tuple artifacts is two-fold:

[7] https://www.gartner.com/reviews/market/business-process-management-platforms

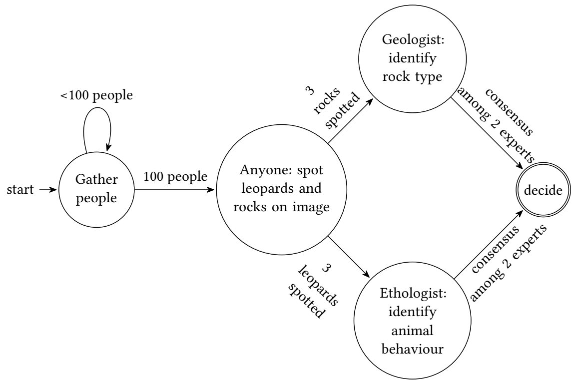

Figure 2: Spotting and classifying leopards and rocks with various expertise levels

Listing 3: Macro-task description

```
launch → count
    guard: none
    actions:
        -- code from Listing 1

count → decide
    guard: task 1 has 10 answers
    actions:
        prepare task 2 as computation
        take skill-weighted-majority(answers 1)

decide: final

result: answers 2
```

first they offer a data-oriented perspective of both crowd contributions and the data aggregation life-cycle, without relying on an external procedural language. Second, this formalism enables static analysis and formal verification, even in the presence of data.

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

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

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.

Since guards and actions can be defined completely with queries on the Headwork relational schema, and since any number of states can be envisioned, a wide set of task compositions can be expressed: sequences of questions, conditional branching, loops. 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 2 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.

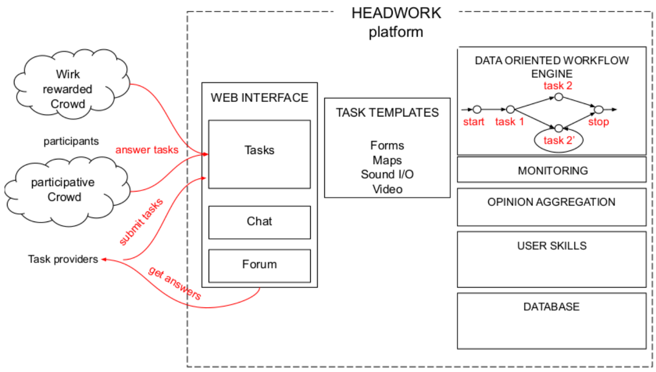

Figure 3: The Headwork architecture

4 THE HEADWORK PLATFORM

The platform, written in PHP (around 7,000 lines) is organized as follows (Figure 3). The main challenges we faced was to devise a fully relational abstraction for user interaction (by contrast with typical MVC approaches) and to capture the relevant crowd primitives. More precisely, *Task providers* submit a job as a JSON file encoding the tuple artifact, in the SQL or Crowdy language, based on the various available templates (HTML forms, Maps, Sound I/O, custom Javascript, ...). The workflow engine then processes the automaton and render tasks to participants through the *Web interface* (Javascript, Bootstrap). Participants can create an account, give their profile (skills), see the list of available tasks ranked according to their skills, and start contributing. As we focus on participative crowdsourcing, tasks can be freely chosen by participants. A task designer can implement a finer participant selection based on skills or gold questions to avoid spammers. If required, Headwork is compatible with a rewarded pool of participants through the Wirk[8] service, to speed-up macro-tasks that could not wait for benevolent participants.

5 DEMO SCENARIO

The demo will start with a basic crowdsourcing interaction for image annotation. Participants are invited to list some skills and annotate a wildlife image, and see how decision are made using majority voting. Then we will demonstrate the flexibility of the interface with controlled text input, HTML forms, maps and

[8] https://wirk.io/en

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[9], with its source code[10] and a companion video[11].

6 CONCLUSION AND FUTURE WORK

In this demo, we presented HEADWORK, an open-source crowdsourcing platform. 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 interns from Rennes University that contributed to this project. This work was also funded by the French National Research Agency grant HEADWORK[12] (ANR-16-CE23-0015).

REFERENCES

[1] Salman Ahmad, Alexis Battle, Zahan Malkani, and Sepander Kamvar. 2011. The Jabberwocky Programming Environment for Structured Social Computing. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST '11)*. ACM, New York, NY, USA, 53–64. https://doi.org/10.1145/2047196.2047203

[2] Pierre Bourhis, Loïc Hélouët, Zoltán Miklós, and Rituraj Singh. 2020. Data Centric Workflows for Crowdsourcing. In *Application and Theory of Petri Nets and Concurrency - 41st International Conference, PETRI NETS 2020, Paris, France, June 24-25, 2020, Proceedings (Lecture Notes in Computer Science)*, Ryszard Janicki, Natalia Sidorova, and Thomas Chatain (Eds.), Vol. 12152. Springer, 24–45. https://doi.org/10.1007/978-3-030-51831-8_2

[3] Chengliang Chai, Ju Fan, Guoliang Li, Jiannan Wang, and Yudian Zheng. 2019. Crowdsourcing database systems: Overview and challenges. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 2052–2055.

[4] Zhao Chen, Rui Fu, Ziyuan Zhao, Zheng Liu, Leihao Xia, Lei Chen, Peng Cheng, Caleb Chen Cao, Yongxin Tong, and Chen Jason Zhang. 2014. gMission: A general spatial crowdsourcing platform. *Proceedings of the VLDB Endowment* 7, 13 (2014), 1629–1632.

[5] Seth Cooper, Firas Khatib, Adrien Treuille, Janos Barbero, Jeehyung Lee, Michael Beenen, Andrew Leaver-Fay, David Baker, Zoran Popović, and Foldit players. 2010. Predicting protein structures with a multiplayer online game. *Nature* 466, 7307 (2010), 756–760.

[6] Alin Deutsch, Richard Hull, Fabio Patrizi, and Victor Vianu. 2009. Automatic Verification of Data-Centric Business Processes. In *Proceedings of the 12th International Conference on Database Theory (ICDT '09)*. Association for Computing Machinery, New York, NY, USA, 252–267. https://doi.org/10.1145/1514894.1514924

[7] Lucy Fortson, Karen Masters, Robert Nichol, EM Edmondson, C Lintott, J Raddick, and J Wallin. 2012. Galaxy zoo. *Advances in machine learning and data mining for astronomy* 2012 (2012), 213–236.

[8] Kosetsu Ikeda, Atsuyuki Morishima, Habibur Rahman, Senjuti Basu Roy, Saravanan Thirumuruganathan, Sihem Amer-Yahia, and Gautam Das. 2016. Collaborative crowdsourcing with crowd4u. *Proceedings of the VLDB Endowment* 9, 13 (2016), 1497–1500.

[9] Vassillis-Javed Khan, Konstantinos Papangelis, Ioanna Lykourentzou, and Panos Markopoulos. 2019. Macrotask Crowdsourcing. *Cham: Springer International Publishing* (2019).

[10] Guoliang Li, Jiannan Wang, Yudian Zheng, Ju Fan, and Michael J Franklin. 2018. Crowdsourced data management. *Hybrid Human-Machine Data Management* (2018).

[11] Greg Little, Lydia B. Chilton, Max Goldman, and Robert C. Miller. 2010. TurKit: Human Computation Algorithms on Mechanical Turk. In *Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology (UIST '10)*. ACM, New York, NY, USA, 57–66. https://doi.org/10.1145/1866029.1866040

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

[13] Nelsi Melgarejo and Luis Camacho. 2018. Implementation of a web platform for the preservation of american native languages. In *2018 IEEE XXV International Conference on Electronics, Electrical Engineering and Computing (INTERCON)*. IEEE, 1–4.

[14] Robert Simpson, Kevin R Page, and David De Roure. 2014. Zooniverse: observing the world's largest citizen science platform. In *Proceedings of the 23rd international conference on world wide web*. 1049–1054.

[9] https://headwork.irisa.fr
[10] https://gitlab.inria.fr/druid-public/headwork
[11] https://headwork.irisa.fr/headwork-demo.mp4
[12] https://headwork.irisa.fr/headwork-web/