An Adaptive Web Interface for Microtask-Based Crowdsourcing

Ilya Sukhopluev
Ural Federal University
Yekaterinburg, Russia
suhoy95@gmail.com

Dmitry Ustalov
Ural Federal University
Yekaterinburg, Russia
dmitry.ustalov@urfu.ru

***Abstract*—In unpaid microtask-based crowdsourcing, it is of great importance to lower the worker entrance barrier for attracting more volunteers and keeping their motivation high. To do that, the worker interfaces should be easy to use. In this paper, we explore the three-tier software architecture for running microtasks and also present Boyarin, an open source front-end Web application for bridging the gap between the crowd workers and the underlying Mechanical Tsar crowdsourcing engine.**

I. Introduction

Microtask-based crowdsourcing implies submission of the human intelligence tasks (HITs) to the controlled or uncontrolled crowd of human workers. This approach for hybrid human-machine computation was popularized by Amazon MTurk [1], an online labor marketplace when the workers receive monetary incentives for the submitted answers. A number of paid crowdsourcing platforms offer convenient, yet proprietary task design frameworks, such in the cases of CrowdFlower [2] and Yandex.Toloka [3].

In contrast, unpaid crowdsourcing relies on open source software to run the annotation process and collect the workers' answers [4]. To make it more convenient for the volunteers, the worker interface should be adaptive both to the devices from which the workers do the annotation and to their behavioural patterns. The latter aspect needs to be properly captured and analyzed in a post-hoc manner [5].

In this paper, we are focusing on the software engineering aspect of microtask-based crowdsourcing. We discuss the existent systems in Section II. Then, we emphasize on the three-tier architecture described in Section III and present Boyarin, a flexible Web-based annotation front-end, in Section IV. Finally, we show illustrative examples in Section V and provide concluding remarks in Section VI. The software we demonstrate is open source and is available under a libré license.

II. Related Work

Several frameworks are being available, such as TurKit [6], Bossa [7], PyBossa [8], psiTurk [9], Troia [10], and Mechanical Tsar [11]. Some, like TurKit and psiTurk, are tighty coupled with the paid MTurk. Others, like Bossa and Troia, seems to be not extensively maintained today. PyBossa, despite its impressive features and flexibility, lacks the statistical quality control mechanisms offered by Mechanical Tsar.

III. Architecture

In the three-tier architecture, depicted at Fig. 1, the crowdsourcing system is composed of three primary subsystems: the *database* storing the task, worker and answer information, the *back-end* (or the Mechanical Tsar *engine*) dealing with task allocation and answer aggregation, and the *front-end* representing the tasks to the workers and collecting back their answers. In our setup, the engine provides the RESTful API over HTTP or HTTPS to prevent the workers interacting with the engine directly. Additionally, the front-end communicates with the *telemetry system* that tracks the worker activity for gathering the implicit feedback from them.

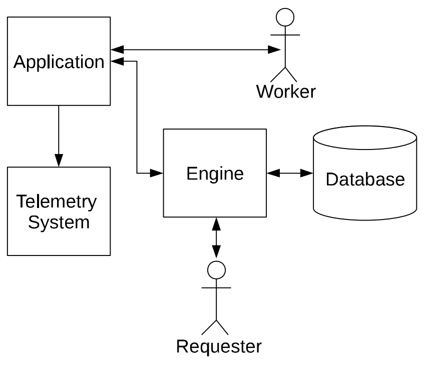


Fig. 1. The architectural diagram representing the front-end (the application), the back-end (the engine) and the supplementary systems along with the worker and the requester users

The engine also exposes the graphical user interface for making it possible for the requester to tune the annotation parameters and perform data management operations. According to the present architecture, the front-end system performs three primary activities: (1) presenting the list of the available annotation stages, (2) rendering the task from the stage chosen by the worker, and (3) receiving the answers submitted by the worker. An annotation stage is the set of tasks grouped by the similar objective, like "select the similar words" or "confirm whether the message is a spam or ham".

Although the engine stores the worker metadata, it does not deal with the worker authentication, so this functionality should be implemented by the front-end. Under this setting, the front-end may implement any authentication mechanism possible, e.g., social network login, anonymous login, etc., but the engine will manipulate with the internal worker identifiers by securing their personal identities (Fig. 2).

ISBN 978-952-68397-8-3 (PDF)

Since that the answer processing is done by the Mechanical Tsar engine, the front-end system has no possibility to perform formal answer validation, i.e., the presence of the answer in the single choice questions, etc. Instead, it is still possible to communicate with the engine that produces the validation errors in the machine-readable form. Thus, the role of the front-end in this system is to make interacting with the engine accessible to humans (Fig. 3).

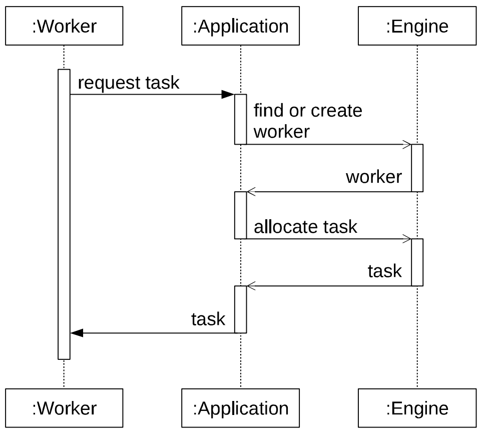


Fig. 2. Sequence diagram for task allocation

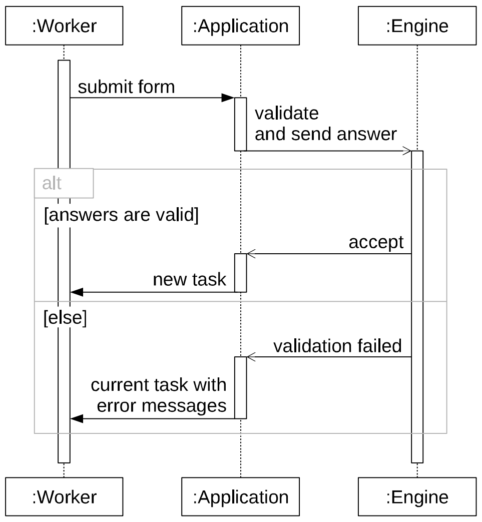


Fig. 3. Sequence diagram for answer submission

IV. Implementation

Boyarin, our implementation of the front-end system [12], is a Web application written in JavaScript using the Node.js runtime environment [13] and specialized middleware. The authentication routines are available for Facebook, VK, GitHub, or "login-less" approach that records only the worker's IP address and browser identifier combination. The worker sessions are stored using encrypted HTTP cookies, which are small fragments of data stored in the browser for the specified time.

The interaction with the engine is asynchronous due to the properties of the Node.js runtime. For encapsulating this interaction, all the requests are wrapped into the `Connector`

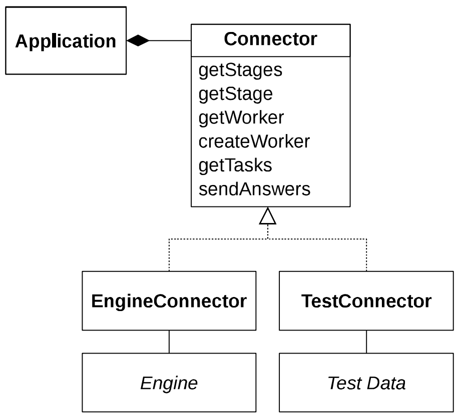


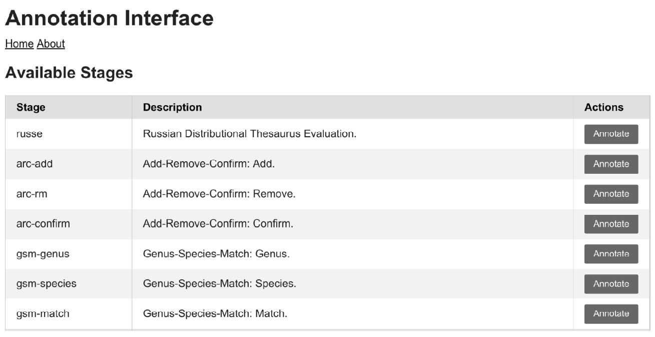
Fig. 4. Connector, the wrapper object

object implementing the Proxy design pattern (Fig. 4). This also simplifies unit testing by providing the mock data without any need for network connection during the tests.

For each task, represented as an annotation stage in Boyarin and Mechanical Tsar, it is possible to design the specific worker interface template, but the requester has also been provided with several generic options being available. If the customization is needed, the requester chooses the authentication approach and makes relatively small changes to the present HTML, CSS and JavaScript code snippets. A special attention is paid to the convenience of the users and the compatibility to mobile devices by automatically rearranging the worker interface according to the screen resolution of the particular device. For gathering the telemetry information, we use the Piwik open source Web analytics system hosted on our servers [14].

V. Examples

The above-mentioned worker interface has been successfully used in the evaluation task for the Russian Distributional Thesaurus [15]. For instance, Fig. 5 shows the list of available annotation stages and Fig. 6 shows the tasks representation for the selected stage. The requester has a separate user interface of the Mechanical Tsar engine for tuning the annotation process and downloading and uploading the data as well (Fig. 7).

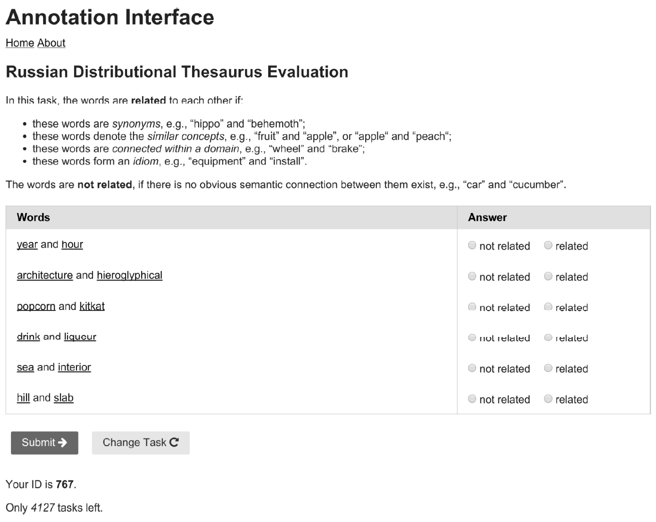

Annotation Interface

Home About

Available Stages

Stage	Description	Actions
russe	Russian Distributional Thesaurus Evaluation.	Annotate
arc-add	Add-Remove-Confirm: Add.	Annotate
arc-rm	Add-Remove-Confirm: Remove.	Annotate
arc-confirm	Add-Remove-Confirm: Confirm.	Annotate
gsm-genus	Genus-Species-Match: Genus.	Annotate
gsm-species	Genus-Species-Match: Species.	Annotate
gsm-match	Genus-Species-Match: Match.	Annotate

Fig. 5. List of the available annotation stages (interface captions are translated for convenience of readers into English; originally all the interface elements are in Russian)



Annotation Interface

Home About

Russian Distributional Thesaurus Evaluation

In this task, the words are **related** to each other if:

- these words are *synonyms*, e.g., “hippo” and “behemoth”;
- these words denote the *similar concepts*, e.g., “fruit” and “apple”, or “apple“ and “peach“;
- these words are *connected within a domain*, e.g., “wheel” and “brake”;
- these words form an *idiom*, e.g., “equipment” and “install”.

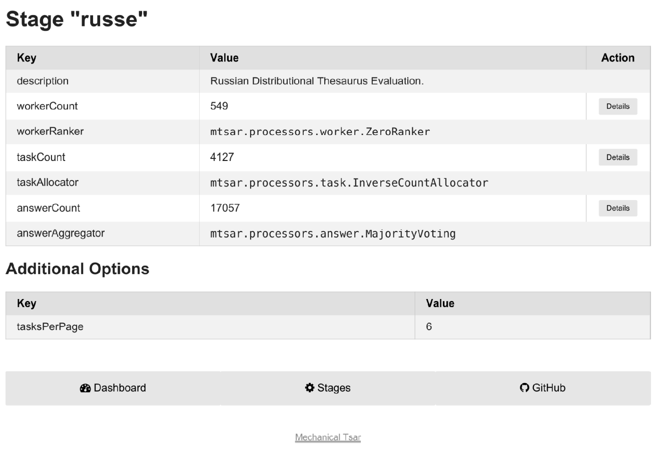
The words are **not related**, if there is no obvious semantic connection between them exist, e.g., “car” and “cucumber”.

Words	Answer
year and hour	○ not related ○ related
architecture and hieroglyphical	○ not related ○ related
popcorn and kitkat	○ not related ○ related
drink and liqueur	○ not related ○ related
sea and interior	○ not related ○ related
hill and slab	○ not related ○ related

Your ID is **767**.

Only *4127* tasks left.

Fig. 6. Microtasks of the selected annotation stage (interface captions are translated for convenience of readers into English; originally all the interface elements are in Russian)



Stage "russe"

Key	Value	Action
description	Russian Distributional Thesaurus Evaluation.	
workerCount	549	Details
workerRanker	mtsar.processors.worker.ZeroRanker	
taskCount	4127	Details
taskAllocator	mtsar.processors.task.InverseCountAllocator	
answerCount	17057	Details
answerAggregator	mtsar.processors.answer.MajorityVoting	

Additional Options

Key	Value
tasksPerPage	6

Fig. 7. The requester interface offered by the Mechanical Tsar engine

VI. Conclusion

The software developed within this study is publicly available for use and modification [12]. We believe that the release of this software, Boyarin, will facilitate the crowdsourcing studies of other researchers that might benefit of convenient task offering procedure provided by the front-end and the statistical answer aggregation algorithms provided by the Mechanical Tsar engine for increasing the annotation reliability. Since that unpaid crowdsourcing completes slowlier but yields results of similar or higher quality compared to its paid counterpart [4], we plan to focus on providing better crowdsourcing task design in further studies.

Acknowledgment

The reported study was funded by RFBR according to the research project No. 16-37-00354 mol_a. We would like to thank Denis Shirgin for the initial implementation of the annotation front-end. We are also grateful to the four reviewers who offered useful comments on the present paper.

References

[1] Amazon Mechanical Turk - Welcome, Web: https://www.mturk.com/mturk/welcome.

[2] CrowdFlower | People-powered Data Enrichment Platform, Web: https://www.crowdflower.com/.

[3] Yandex.Toloka, Web: https://toloka.yandex.com/.

[4] R.M. Borromeo, T. Motomichi, “An investigation of unpaid crowdsourcing”, *Human-centric Computing and Information Sciences*, vol.6(1), Aug.2016, pp. 1–19.

[5] U. Gadiraju, R. Kawase, S. Dietze, G. Demartini, “Understanding malicious behavior in crowdsourcing platforms: The case of online surveys”, *in Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, Apr. 2015, pp. 1631-1640.

[6] G. Little, L.B. Chilton, M. Goldman, R.C. Miller, “TurKit: human computation algorithms on mechanical turk”, *in Proceedings of the 23nd annual ACM symposium on User interface software and technology*, Oct. 2010, pp. 56-66.

[7] E.J. Korpela, “SETI@home, BOINC, and Volunteer Distributed Computing”, *Annual Review of Earth and Planetary Sciences*, vol.40(1), May.2012, pp. 69-87.

[8] The ultimate crowdsourcing framework - PyBossa, Web: http://pybossa.com/.

[9] T.M. Gureckis, J. Martin, J. McDonnell, A.S. Rich, D. Markant, A. Coenen, D. Halpern, J.B. Hamrick, P. Chan, “psiTurk: An open-source framework for conducting replicable behavioral experiments online”, *Behavior Research Methods*, vol.48(3), Oct.2015, pp. 829–842.

[10] ipeirotis/Troia-Server: Quality Control API for Crowdsourcing Applications, Web: https://github.com/ipeirotis/Troia-Server.

[11] D.A. Ustalov, “A Crowdsourcing Engine for Mechanized Labor”, *in Proceedings of the Institute for System Programming*, vol.27(3), Jul.2015, pp. 351-364.

[12] mtsar/boyarin: A servant of Mechanical Tsar, Web: https://github.com/mtsar/boyarin.

[13] Node.js, Web: https://nodejs.org/.

[14] Free Web Analytics Software, Web: https://piwik.org/.

[15] A. Panchenko, D. Ustalov, N. Arefyev, D. Paperno, N. Konstantinova, N.V. Loukachevitch, C. Biemann, “Human and machine judgements for Russian semantic relatedness”, Springer CCIS, in press.