Teleboyarin—Mechanized Labor for Telegram

Dmitry Ustalov*†
*IMM UB RAS, Yekaterinburg, Russia
†Ural Federal University, Yekaterinburg, Russia
dau@imm.uran.ru

***Abstract*—A successful crowdsourced annotation is a joint result of good task design, high worker motivation and appropriate aggregation methods. Having no access to global online labor marketplaces complicates inviting a large amount of workers, making it highly topical to provide those potential ones with the lowest possible entry barrier for joining the annotation. This paper presents an instant messaging bot that is designed for rapid delivery of annotation processes powered by the Mechanical Tsar mechanized labor engine over the Telegram messaging system.**

I. Introduction

Performance of the crowdsourced annotation may benefit from new techniques of work organization including inter-worker communication and career ladders [1]. In developing countries, which still have access to global online marketplaces like Amazon Mechanical Turk (AMT) [2], workers prefer to get paid in some form of a commodity instead of money [3]. Since that there are virtually no Russian speaking workers existing on AMT [4], it seems to be reasonable to put a focus on *altruistic crowd work*, in which the annotation is performed by volunteers for non-monetary rewards. However, the volunteers should be motivated to support a particular crowd project. This is achieved when either the deliverables are useful for them, or the annotation process is attractive and easy to access [5]. This paper presents Teleboyarin—an open source instant messaging bot designed for crowd annotation.

II. Related Work

Annotation interfaces have received significant attention in last years: researchers experimented with various forms of human-computer interaction including smartphone applications and online social network applications. Nevertheless, the discussion of better task delivery is still active.

Yan et al. presented mCrowd, a proxy for accessing human intelligence tasks on AMT from mobile devices [6]. Difallah et al. demonstrated OpenTurk, a Facebook application for embedding annotation processes into the social network and avoiding external authentication [7]. Matera et al. proposed an easy to use crowdsourcing task manager for mechanized labor annotation workflows [8].

Yimam et al. created WebAnno, a self-sufficient Web-based distributed annotation tool [9]. Bocharov et al. developed a corpus annotation tool relying on OAuth for inviting workers from social networks [10]. Braslavski et al., inspired by this approach, applied it for annotating a lexical resource [11].

III. Mechanical Tsar

Mechanical Tsar is an open source crowdsourcing engine designed for rapid deployment of mechanized labor workflows [12], [13]. This engine imposes a three-tier architecture composed of (1) the *database* system that stores the task, worker data and their answers, (2) the back-end *engine* that deals with adaptive task allocation, worker ranking and answer aggregation, and (3) the front-end *application* that provides workers with the corresponding user interface. Thus, a worker interacts only with the application, which communicates with the underlying engine RESTful API only if it is necessary (Fig. 1). Such an architecture makes it possible to detach the presentation layer from the computation logic layer and thus provide the workers with highly customized user interfaces for specific needs, e.g. for mobile annotation, Web-based annotation, etc. In Mechanical Tsar, the entities (tasks, workers and answers) are grouped into *processes* and isolated from each other. However, it is possible to attach arbitrary textual tags to any entity for distinguishing simplification.

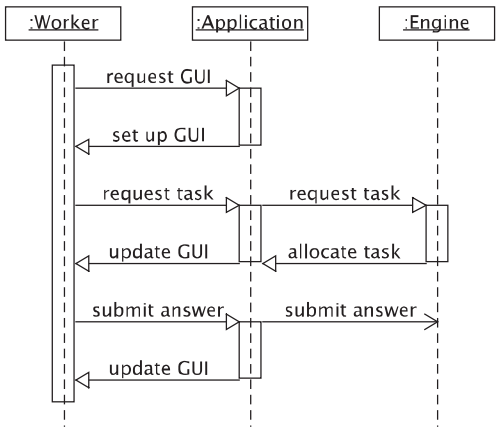


Fig. 1. Sequence diagram of the annotation process

IV. Telegram Bot

Telegram is a cross-platform instant messaging system that provides a convenient Bot API for building chatbots, which interact with a user or with a group of users [14]. The API provides additional methods for augmenting communication, e.g. rich text formatting, input completion, command lists, etc.

The three-tier architecture of Mechanical Tsar requires annotation bot to implement the *application* layer, i.e. it should identify users, provide them with tasks, receive answers from them and deliver the corresponding data from and to the engine over its API. Since all the Telegram users get a unique user identifier (ID) verified by a cell phone number, it is sufficient to represent the users with the above mentioned IDs in order to eliminate the identification problem.

ISBN 978-952-68397-0-7 (paperback), ISBN 978-952-68397-1-4 (PDF)

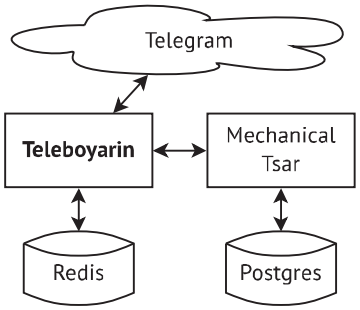


Fig. 2. Architecture of the Telegram bot

Given the fact that annotation is a sequential process, i.e. a worker requests a task, then submits answers, then gets a new task, etc., it is necessary to maintain the state of workers—if the bot is waiting for the answer for the given task or if it is idle w.r.t. the particular worker. The architecture of the bot is present at Fig. 2: the bot communicates with Telegram users via pooling or webhooks, keeps its state in the external Redis cache for recovering between potential restarts, and interacts with Mechanical Tsar over its RESTful API. Table I represents the command system, i.e. the textual user interface of the bot. In order to response to sequential commands correctly, the bot has been implemented as a finite state machine with the state diagram presented at Fig. 3.

TABLE I. BOT COMMANDS

Command	Description	State
`/start`	Ask the bot to send a greeting.	Idle
`/annotate`	Start the annotation of a process.	Idle
`/processes`	List all the annotation processes.	Idle
`/process`	Show the status of a process.	Idle
`/version`	Display the engine version.	Idle
`/cancel`	Cancel any current operation.	Any
`/stop`	Stop the annotation.	Answer

V. EXPERIMENTS

At the present moment, the evaluation process for finding potential bugs and receiving the worker feedback is ongoing. A typical chat session with the bot is going as follows. A worker initiates a conversation with the bot (the `/start` message is sent automatically) and then sends a command from Table I that affect the bot state. In case the bot requires a special kind of answer, it provides the worker with a convenient input method. An illustrative example of messaging-based annotation is present at Fig. 4. Firstly, a worker initiates the conversation and asks the bot to list the processes available for annotation using the `/annotate` command. This command transits the bot state from the initial *Idle* state to *Annotate* specifying that the next message received from the user should be either the process name or the `/cancel` command (Fig. 4a). Having received the process name, the bot requests a task from the engine and then forwards it to the worker (Fig. 4b), while additionally switching the state to *Answer* to wait for an answer. When the answer is received, the bot requests the engine to store it, confirming the worker that the answer has been saved, and then transits to the *Annotate* state, again (Fig. 4c). The worker may choose the `/stop` answer, which turns the bot into the *Idle* state (Fig. 4d).

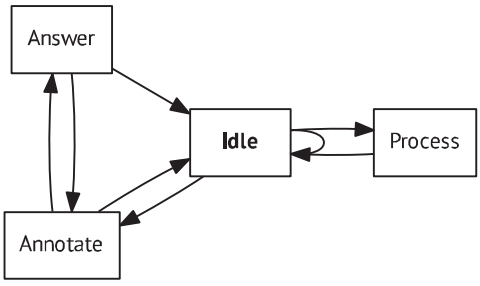


Fig. 3. State diagram of the bot

VI. CONCLUSION

In the future, microtasks will substitute such jobs as search assessors and the similar ones. Hence, the more available microtasks there are, the more people will benefit from their execution. However, there are several interesting reasons for further work.

Games. Games with a purpose is a highly attractive crowdsourcing genre and using instant messaging may facilitate developing fun, yet useful, text-based games involving one or several workers. The main challenge is implementation, which requires real-time interaction between the player(s) and the game coordinator.

Co-op. Since that even a simple implementation of collaborative annotation does increase the worker activity [15], multi-worker annotation is a topical kind of study. The main challenge is in implementing the desired functionality without breaking the API usage constraints.

Rewards. Modern messengers like Facebook Messenger and MoneyTalk support money transfers between users. The main challenge of implementing this kind of reward is local legislation issues, especially the anti-fraud laws.

Source code of Teleboyarin is available on GitHub under the MIT license [16]. Indeed, similar functionality could be implemented in other messaging systems like Jabber, Slack, WhatsApp, etc.

ACKNOWLEDGMENT

This work is supported by the Russian Foundation for the Humanities, project no. 13-04-12020 “New Open Electronic Thesaurus for Russian”.

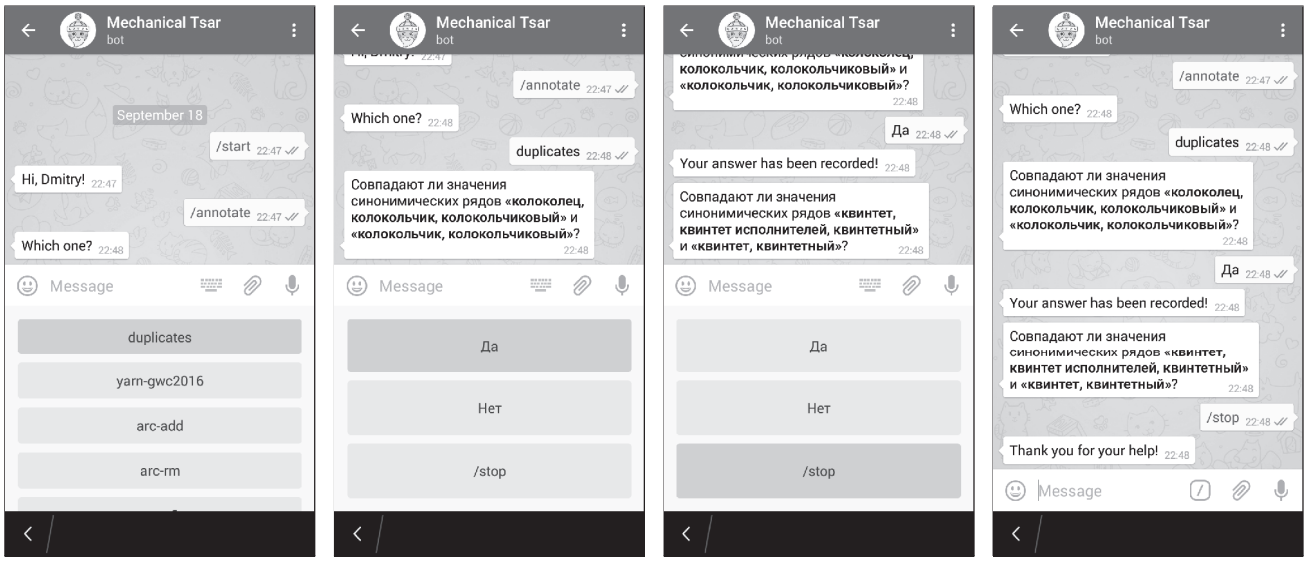
REFERENCES

[1] A. Kittur, J.V. Nickerson, M. Bernstein, et al., “The Future of Crowd Work”, *in Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, Feb. 2013, pp. 1301-1318.

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

[3] N. Samdaria, A. Mathur, R. Balakrishnan, “Paying in Kind for Crowdsourced Work in Developing Regions”, *in Pervasive Computing: 10th International Conference, Pervasive 2012*, Jun. 2012, pp. 343-360.

[4] E. Pavlick, M. Post, A. Irvine, D. Kachaev, C. Callison-Burch, “The Language Demographics of Amazon Mechanical Turk”, *Transactions of the Association for Computational Linguistics*, vol.2, Feb. 2014, pp. 79-92.



(a) Selecting a process to annotate (b) Choosing an answer for the task (c) Deciding to stop for now (d) Being appreciated

Fig. 4. Annotating with Teleboyarin

[5] B. Rahmanian, J.G. Davis, "User Interface Design for Crowdsourcing Systems", *in Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces*, May 2014, pp. 405-408.

[6] T. Yan, M. Marzilli, R. Holmes, et al., "mCrowd: A Platform for Mobile Crowdsourcing", *in Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, Nov. 2009, pp. 347-348.

[7] D.E. Difallah, G. Demartini, P. Cudré-Mauroux, "Pick-A-Crowd: Tell Me What You Like, and I'll Tell You What to Do", *in Proceedings of the 22Nd International Conference on World Wide Web*, May 2013, pp. 367-374.

[8] T. Matera, J. Jakes, M. Cheng, S. Belongie, "A User Friendly Crowdsourcing Task Manager", *in Workshop on Computer Vision and Human Computation (CVPR)*, Jun. 2014.

[9] S.M. Yimam, I. Gurevych, et al., "WebAnno: A Flexible, Web-based and Visually Supported System for Distributed Annotations", *in Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, Aug. 2013, pp. 1-6.

[10] V. Bocharov, S. Alexeeva, D. Granovsky, E. Protopopova, M. Stepanova, A. Surikov. "Crowdsourcing morphological annotation", *in Computational Linguistics and Intellectual Technologies: papers from the Annual conference "Dialogue"*, Jun. 2013, pp. 109-124.

[11] P. Braslavski, D. Ustalov, M. Mukhin, "A Spinning Wheel for YARN: User Interface for a Crowdsourced Thesaurus", *in Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, Apr. 2014, pp. 101-104.

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

[13] Mechanical Tsar, Web: http://mtsar.nlpub.org/.

[14] Telegram Bot API, Web: https://core.telegram.org/bots/api.

[15] D. Ustalov, "Towards crowdsourcing and cooperation in linguistic resources", Springer CCIS, in press.

[16] mtsar/teleboyarin, Web: https://github.com/mtsar/teleboyarin.