A Crowdsourcing Engine for Mechanized Labor

D.A. Ustalov <dau@imm.uran.ru>,
Institute of Mathematics and Mechanics of Ural Branch of Russian Academy of Sciences, 16 Sofia Kovalevskaya Str., Yekaterinburg, 620990, Russian Federation

Abstract. Microtask crowdsourcing implies decomposing a difficult problem into smaller pieces. For that a special human-computer platform like CrowdFlower or Amazon Mechanical Turk is used to submit tasks for human workers motivated by either micropayments or altruism to solve. Examples of successful crowdsourcing applications are food nutrition estimation, natural language processing, criminal invasion detection, and other problems so-called “AI-hard”. However, these platforms are proprietary and requiring additional software for maintaining the output quality. This paper presents the design, architecture and implementation details of an open source engine for executing microtask-based crowdsourcing annotation stages. The engine controls the entire crowdsourcing process including such elements as task allocation, worker ranking, answer aggregation, agreement assessment, and other means for quality control. The present version of the software is implemented as a three-tier system, which is composed of the application level for the end-user worker interface, the engine level for the Web service controlling the annotation process, and the database level for the data persistence. The RESTful API is used for interacting with the engine. The methods for controlling the annotation are implemented as processors that are initialized using the dependency injection mechanism for achieving the loose coupling principle. The functionality of the engine has been evaluated by both using unit tests and replication of a semantic similarity assessment experiment.

Keywords: crowdsourcing engine; mechanized labor; human-assisted computation; task allocation; worker ranking; answer aggregation

DOI: 10.15514/ISPRAS-2015-27(3)-25

For citation: Ustalov D.A. A Crowdsourcing Engine for Mechanized Labor. *Trudy ISP RAN/Proc. ISP RAS, vol. 27, issue 3, 2015*, pp. 351-364. DOI: 10.15514/ISPRAS-2015-27(3)-25.

1. Introduction

Nowadays, crowdsourcing is a popular and a very practical approach for producing and analyzing data, solving complex problems that can be splitted into many simple

and verifiable tasks, etc. Amazon's MTurk[1], a well known online labor marketplace, promotes crowdsourcing as the *artificial artificial intelligence*.

In the *mechanized labor* genre of crowdsourcing, a requester submits a set of tasks that are solved by the crowd workers on the specialized platform. Usually, the workers receive micropayments for their performance; hence, it is of high interest to reach the happy medium between the cost and the quality. The work, as described in this paper, presents an engine for controlling a crowdsourcing process.

The rest of this paper is organized as follows. Section 2 reviews the related work. Section 3 defines the problem of lacking the control software for crowdsourcing. Section 4 presents a two-layer approach for crowdsourcing applications separating the engine from the end-user application. Section 5 describes the implementation of such an engine. Section 6 briefly evaluates the present system. Section 7 concludes with final remarks and directions for the future work.

2. Related Work

There are several approaches for controlling the entire crowdsourcing process.

Whitehill et al. proposed the *GLAD*[2] model that, for the first time, connects such variables as task difficulty, worker experience and answer reliability for image annotation [1].

Bernstein et al. created the *Soylent* word processor, which automatically submits text formatting and rewriting tasks to the crowd on MTurk [2]. The paper also introduces the *Find-Fix-Verify* workflow, which had highly influenced many other researchers in this field of study.

Demartini, Difallah & Cudré-Mauroux developed *ZenCrowd*, another popular approach for controlling crowdsourcing, which was originally designed for mapping the natural language entities to the Linked Open Data [3]. ZenCrowd is based on the EM-algorithm and deploys the tasks to MTurk.

The idea of providing an integrated framework for a crowdsourcing process is not novel and has been addressed by many authors both in academia and the industry, e.g. WebAnno [4], OpenCorpora [5] and Yet Another RussNet [6].

However, the mentioned products are problem-specific and using them for crowdsourcing different tasks may be non-trivial. Moreover, that software do often force the only possible approach for controlling the process of crowdsourcing, which in some cases may result in suboptimal performance.

2.1 Task Allocation

Lee, Park & Park created a dynamic programming method for task allocation among workers showing that consideration of worker's expertise increases the output quality [7].

[1] http://mturk.com/
[2] http://mplab.ucsd.edu/~jake/

Yuen, King & Leung used probabilistic matrix factorization to allocate tasks in the similar manner that recommender systems do [8].

Karger, Oh & Shah proposed a budget-optimal task allocation algorithm inspired by belief propagation and low-rank matrix approximation being suitable for inferring correct answers from those submitted by the workers [9].

2.2 Worker Ranking

Welinder & Perona presented an online algorithm for estimating annotator parameters that requires expert annotations to assess the performance of the workers [10].

Difallah, Demartini & Cudré-Mauroux used social network profiles for determining the worker interests and preferences in order to personalize task allocation [11].

Daltayanni, de Alfaro & Papadimitriou developed the *WorkerRank* algorithm for estimating the probability of getting a job on the *oDesk* online labor marketplace utilizing employer implicit judgements [12].

2.3 Answer Aggregation

The answers are often aggregated with majority voting, which is highly efficient for small number of annotators per question [9]. Some works use a fixed number of answers to aggregate [5].

Sheshadri & Lease released SQUARE[3], a Java library containing implementations of various consensus methods for crowdsourcing [13], i.e. such methods as ZenCrowd [3], majority voting, etc.

Meyer et al. developed *DKPro Statistics*[4] implementing various popular statistical agreement, correlation and significance analysis methods that can be internally used in answer aggregation methods [14].

2.4 Cost Optimization

Satzger et al. presented an auction-based approach for crowdsourcing allowing workers to place bids on relevant tasks and receive payments for their completion [15].

Gao & Parameswaran proposed algorithms to set and vary task completion rewards over time in order to meet the budget constraints using Markov decision processes [16].

Tran-Thanh et al. developed the *Budgeteer* algorithm for crowdsourcing complex workflows under budget constraints that involves inter-dependent micro-tasks [17].

[3] http://ir.ischool.utexas.edu/square/

[4] https://code.google.com/p/dkpro-statistics/

3. Related Work

Hosseini et al. defines the four pillars of crowdsourcing making it possible to represent the crowdsourcing system C as the following quadruple [18]:

$$C = (W, R, T, P). \quad (1)$$

Here, W is the set of workers who benefit from their participation in the process C, R is the task requester who benefits from the crowd work deliverables, T is the set of human intelligence tasks provided by the requester R, and P is the crowdsourcing platform that connects these elements.

Unfortunately, there is no open and customizable software for controlling C. This problem is highly topical since using MTurk, the largest crowdsourcing platform, is not possible outside the U.S. making it interesting to develop an independent substitution that can be hosted.

4. Approach

The reference model of a typical mechanized labor crowdsourcing process is present at Fig. 1 and consists of the following steps repeated until either convergence is achieved or the requester stops the process:

1. a *worker* requests a *task* from the *system*,
2. the *system* allocates a *task* for that *worker*,
3. the *worker* submits an *answer* for that *task*,
4. the *system* receives and aggregates the *answer*,
5. the *system* updates the *worker* and *task* parameters.

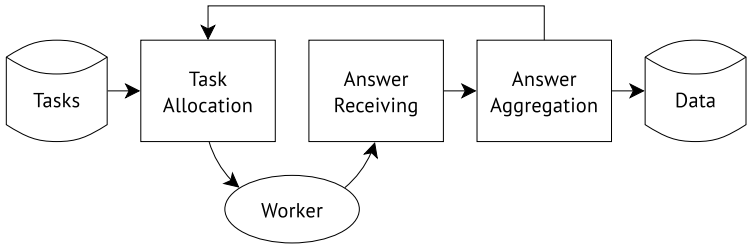

Fig. 1. Reference Model

4.1 Use Case Diagram

Modern recommender systems like PredictionIO[5] and metric optimization tools like MOE[6] separate the *application* layer from the *engine* layer to simplify integration into the existent systems. In crowdsourcing, it is possible to separate the worker

[5] http://prediction.io/

[6] https://github.com/Yelp/MOE

annotation interface (the application) and the crowdsourcing control system (the engine) for the same reason.

The use case diagram present at Fig. 2 shows two actors—the requester and the application—interacting with the engine. The application works with the engine through the specialized programming interface (API) and the requester works with the engine using the specialized graphical user interface (GUI).

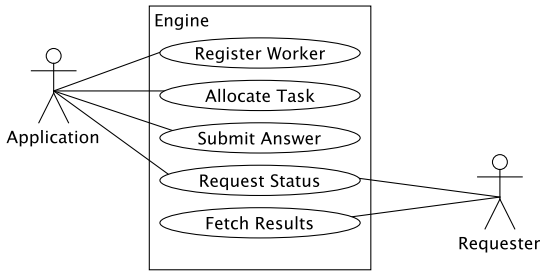

Fig. 2. UML Use Case Diagram

4.2 Sequence Diagram

The sequence diagram at Fig. 3 shows the interaction between those elements: a worker uses the end-user application that is connected to the engine that actually controls the process and provides the application with the appropriate data.

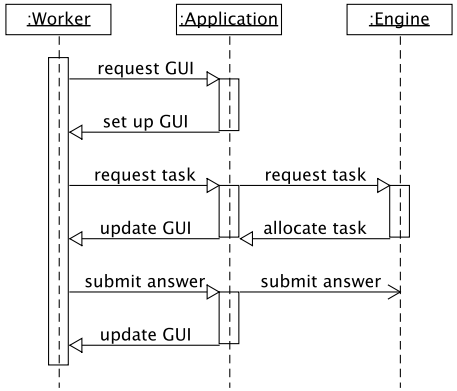

Fig. 3. UML Sequence Diagram

5. Implementation

The proposed system is implemented in the Java programming language as a RESTful Web Service using such APIs as JAX-RS[7] within the Dropwizard[8] framework. The primary data storage is PostgreSQL[9], a popular open source object-relational database.

5.1 Class Diagram

The class diagram at Fig. 4 represents the crowdsourcing system as according to the equation 1. The `Process` class defines a system C and specifies how its elements W, T and A should be processed by the corresponding implementations of these interfaces.

Particularly, an actual processor inherits that abstract class and implements one or many of the following interfaces: `WorkerRanker`, `TaskAllocator`, `AnswerAggregator`. The reason for that is the dependency uncertainty of each particular processor implementation that has been approached by the dependency injection mechanism[10].

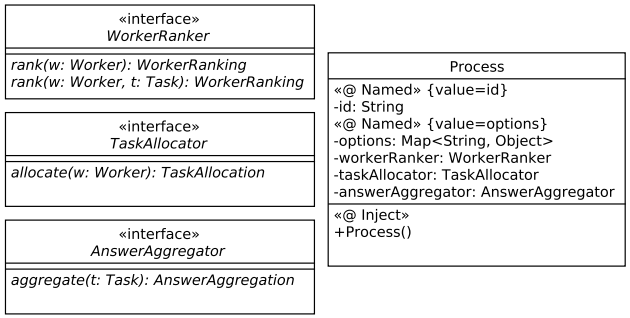

Fig. 4. UML Class Diagram

For example, an implementation of the majority voting technique, which is a popular approach for answer aggregation, should inherit the `AnswerAggregator` interface and provide the implementation of the `aggregate` method that returns an `AnswerAggregation` instance representing the aggregated answer for the given `Task` instance. In order to access the answers stored in the database, the corresponding data access object—`AnswerDAO`—should be injected. Since that the answers cannot be fetched without the correct process identifier, the corresponding

[7] https://jcp.org/en/jsr/detail?id=339
[8] http://dropwizard.io/
[9] http://www.postgresql.org/
[10] https://jcp.org/en/jsr/detail?id=330

`Process` instance should be injected, too. Direct injection of `Process` to `AnswerAggregator` and vice versa causes a circular dependency. The cycle has been successfully broken by injecting a lazily initialized Process provider instead of its actual instance.

On startup, the application configures itself with the provided configuration files, setting up the top-level Guice[11] dependency injector. After establishing a database connection, a database-aware child injector has been created, because it is not possible to achieve during the framework bootstrapping stage. Then, for each defined process, the application initializes a child injector containing process-specific bindings, and that injector is inherited from the database-aware one. Finally, the application exposes these processes by the RESTful API.

5.2 Package Diagram

The system is composed of several packages responsible for its functionality. Since that the Dropwizard framework is used, the most of boilerplate code is already included in the framework. However, such a sophisticated initialization requires additional middleware resulting in the package hierarchy represented at Fig. 5 detailed in Table 1.

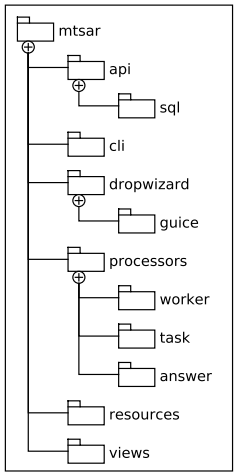

Fig. 5. UML Package Diagram

[11] https://github.com/google/guice

Table 1. Packages

Package	Description
`mtsar`	Utility classes useful to avoid the code repetition.
`mtsar.api`	Entity representations.
`mtsar.api.sql`	Data access objects and object mappers.
`mtsar.cli`	Command-line tools for maintenance and evaluation tasks.
`mtsar.dropwizard`	Middleware for Dropwizard.
`mtsar.processors`	Actual implementations of the methods for controlling workers, tasks, answers.
`mtsar.resources`	Resources exposed by the RESTful API.
`mtsar.views`	View models used by the GUI.

6. Evaluation

The system functionality is tested using JUnit[12]. At the present moment, only classes contained in the `mtsar.processors` and `mtsar.resources` packages are provided with the appropriate unit tests. The continuous integration practice is followed by triggering a build on Travis CI[13] for each change to ensure that all the unit tests have been successfully passed.

In order to make sure the system works, the RUSSE[14] crowdsourced dataset has been used (see [19] for details). The russe process has been configured to use the *zero* worker ranker that simply ranks any worker with zero rank, *inverse count* task allocator that allocates the task with the lowest number of available answers, and the *majority voting* answer aggregator (Fig. 6). Then, the workers, tasks and answers stored in this dataset have been submitted into the system via the RESTful API and the conducted experiment showed that no data have been lost during this activity and the engine does allocate tasks and aggregate answers correctly w.r.t. the chosen processors.

[12] http://junit.org/
[13] https://travis-ci.org/
[14] http://russe.nlpub.ru/

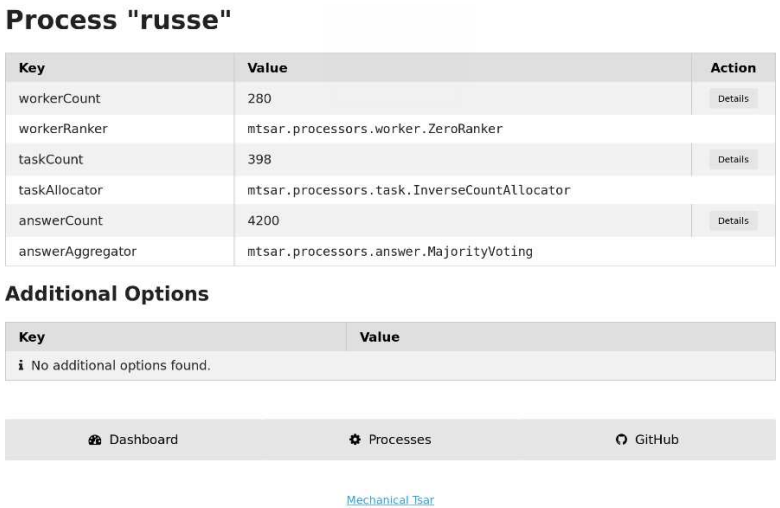

Fig. 6. Graphical User Interface

7. Conclusion

In this study, a crowdsourcing engine for mechanized labor has been presented and described among the used approach and its implementation. Despite the conducted experiment showing promising preliminary results, there are the following reasons for the further work.

Firstly, it is necessary to conduct a field study, which was not possible due to the lack of time. Secondly, it is necessary to integrate state of the art methods for worker ranking, task allocation and answer aggregation into the engine to provide a requester with the best annotation quality at the lowest cost. Finally, it may be useful to extend the engine API and GUI in order to make it more convenient and user-friendly.

The source code of the system is released on GitHub[15] under the Apache License. The documentation is available on GitHub[16] in English and on NLPub[17] in Russian.

Acknowledgements. This work is supported by the Russian Foundation for the Humanities, project № 13-04-12020 “New Open Electronic Thesaurus for Russian”. The author is grateful to the anonymous referees who offered useful comments on the present paper.

[15] https://github.com/dustalov/mtsar
[16] https://github.com/dustalov/mtsar/wiki
[17] https://nlpub.ru/MTsar

References

[1]. J. Whitehill, P. Ruvolo, T. Wu, J. Bergsma, J. Movellan. Whose Vote Should Count More: Optimal Integration of Labels from Labelers of Unknown Expertise. Advances in Neural Information Processing Systems 22. Curran Associates, Inc., 2009, pp. 2035–2043.

[2]. M. S. Bernstein, G. Little, R. C. Miller, B. Hartmann, M. S. Ackerman, D. R. Karger, D. Crowell, K. Panovich. Soylent: A word processor with a crowd inside. Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology (UIST ’10). New York, NY, USA: ACM, 2010, pp. 313–322. doi: 10.1145/1866029.1866078

[3]. G. Demartini, D. E. Difallah, P. Cudré-Mauroux. ZenCrowd: Leveraging Probabilistic Reasoning and Crowdsourcing Techniques for Large-Scale Entity Linking. Proceedings of the 21st International Conference on World Wide Web (WWW ’12). New York, NY, USA: ACM, 2012, pp. 469–478. doi: 10.1145/2187836.2187900

[4]. S. M. Yimam, I. Gurevych, R. E. de Castilho, C. Biemann. 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. Sofia, Bulgaria: Association for Computational Linguistics, 2013, pp. 1–6.

[5]. 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”, vol. 1, no. 12(19). Moscow: RSUH, 2013, pp. 109–124.

[6]. 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. Gothenburg, Sweden: Association for Computational Linguistics, 2014, pp. 101–104.

[7]. S. Lee, S. Park, S. Park. A Quality Enhancement of Crowdsourcing based on Quality Evaluation and User-Level Task Assignment Framework. 2014 International Conference on Big Data and Smart Computing (BIGCOMP). IEEE, 2014, pp. 60–65. doi: 10.1109/BIGCOMP.2014.6741408

[8]. M.-C. Yuen, I. King, K.-S. Leung. TaskRec: A Task Recommendation Framework in Crowdsourcing Systems. Neural Processing Letters, pp. 1–16, 2014. doi: 10.1007/s11063-014-9343-z

[9]. D. R. Karger, S. Oh, D. Shah. Budget-Optimal Task Allocation for Reliable Crowdsourcing Systems. Operations Research, vol. 62, no. 1, pp. 1–24, 2014. doi: 10.1287/opre.2013.1235

[10]. P. Welinder P. Perona. Online crowdsourcing: Rating annotators and obtaining cost-effective labels. 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2010, pp. 25–32. doi: 10.1109/CVPRW.2010.5543189

[11]. D. E. Difallah, G. Demartini, P. Cudré-Mauroux. Pick-A-Crowd: Tell Me What You Like, and I’ll Tell You What to Do. Proceedings of the 22Nd International Conference on World Wide Web (WWW ’13). Rio de Janeiro, Brazil: International World Wide Web Conferences Steering Committee, 2013, pp. 367–374.

[12]. M. Daltayanni, L. de Alfaro, P. Papadimitriou. WorkerRank: Using Employer Implicit Judgements to Infer Worker Reputation. Proceedings of the Eighth ACM International

Conference on Web Search and Data Mining (WSDM '15). New York, NY, USA: ACM, 2015, pp. 263–272. doi: 10.1145/2684822.2685286

[13]. A. Sheshadri, M. Lease. SQUARE: A Benchmark for Research on Computing Crowd Consensus. First AAAI Conference on Human Computation and Crowdsourcing, 2013, pp. 156–164.

[14]. C. M. Meyer, M. Mieskes, C. Stab, I. Gurevych. DKPro Agreement: An Open-Source Java Library for Measuring Inter-Rater Agreement. Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: System Demonstrations. Dublin, Ireland: Dublin City University and Association for Computational Linguistics, 2014, pp. 105–109.

[15]. B. Satzger, H. Psaier, D. Schall, S. Dustdar. Auction-based crowdsourcing supporting skill management. Information Systems, vol. 38, no. 4, pp. 547–560, 2013. doi: 10.1016/j.is.2012.09.003

[16]. Y. Gao, A. Parameswaran. Finish Them! : Pricing Algorithms for Human Computation. Proceedings of the VLDB Endowment, vol. 7, no. 14, 2014. doi: 10.14778/2733085.2733101

[17]. L. Tran-Thanh, T. D. Huynh, A. Rosenfeld, S. D. Ramchurn, N. R. Jennings. Crowdsourcing Complex Workflows under Budget Constraints. Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15). AAAI Press, 2015, pp. 1298–1304.

[18]. M. Hosseini, K. Phalp, J. Taylor, R. Ali. The Four Pillars of Crowdsourcing: a Reference Model. 2014 IEEE Eighth International Conference on Research Challenges in Information Science (RCIS), 2014, pp. 1–12. doi: 10.1109/RCIS.2014.6861072

[19]. A. Panchenko, N. V. Loukachevitch, D. Ustalov, D. Paperno, C. M. Meyer, N. Konstantinova. RUSSE: The First Workshop on Russian Semantic Similarity. Computational Linguistics and Intellectual Technologies: papers from the Annual conference "Dialogue". Moscow: RGGU, 2015, vol. 2, no. 14(21), pp. 89–105.

Инструментарий краудсорсинга для механизированного труда

Д.А. Усталов <dau@imm.uran.ru>,
Институт математики и механики им. Н.Н.Красовского
Уральского отделения Российской академии наук,
620990, г. Екатеринбург, ул. Софьи Ковалевской, д. 16

Аннотация. Краудсорсинг на основе выполнения микрозадач предполагает разделение исходной задачи на множество менее крупных. Микрозадачи выполняются на специализированных человеко-машинных платформах, таких как CrowdFlower и Amazon Mechanical Turk, за что участники процесса краудсорсинга получают некоторое вознаграждение. Среди успешных примеров применения краудсорсинга следует отметить решение задач по оценке калорийности пищи, обработке естественного языка, обнаружению незаконного проникновения на территорию, и

других «ИИ-трудных» задач. Существующие платформы для выполнения микрозадач являются закрытыми; для обеспечения качества результата разметки необходимо предпринимать дополнительные усилия по обработке данных. В данной статье представлен инструментарий для выполнения микрозадач с открытым исходным кодом, проведено описание архитектуры и деталей реализации. Инструментарий управляет всеми аспектами процесса выполнения микрозадач: осуществляет назначение заданий, оценку квалификации участников, агрегацию ответов и оценку их согласованности, а также включает иные подходы к обеспечению качества результата. Текущая версия инструментария реализована в виде трёхзвенной информационной системы, состоящей из уровня приложения с интерфейсом для участников, уровня Веб-сервиса для управления процессом, и уровня хранения данных. Взаимодействие с Веб-сервисом осуществляется при помощи программного интерфейса, построенного на основе архитектурного стиля передачи состояния представления. Методы управления разметкой реализуются в виде процессоров, инициализируемых при помощи механизма внедрения зависимостей для достижения принципа слабой связности системы. Работоспособность инструментария подтверждается наличием модульных тестов и успешным воспроизведением эксперимента по оценке семантической близости слов.

Ключевые слова: краудсорсинг; механизированный труд; человеко-машинные вычисления; назначение заданий; оценка труда участников; агрегация ответов

DOI: 10.15514/ISPRAS-2015-27(3)-25

Для цитирования: Усталов Д.А.. Инструментарий краудсорсинга для механизированного труда. Труды ИСП РАН, том 27, вып. 3, 2015 г., стр. 351-364 (на английском языке). DOI: 10.15514/ISPRAS-2015-27(3)-25.

Список литературы

[1]. J. Whitehill, P. Ruvolo, T. Wu, J. Bergsma, and J. Movellan. Whose Vote Should Count More: Optimal Integration of Labels from Labelers of Unknown Expertise. Advances in Neural Information Processing Systems 22. Curran Associates, Inc., 2009, pp. 2035–2043.

[2]. M. S. Bernstein, G. Little, R. C. Miller, B. Hartmann, M. S. Ackerman, D. R. Karger, D. Crowell, K. Panovich. Soylent: A word processor with a crowd inside. Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology (UIST '10). New York, NY, USA: ACM, 2010, pp. 313–322. doi: 10.1145/1866029.1866078

[3]. G. Demartini, D. E. Difallah, P. Cudré-Mauroux, ZenCrowd: Leveraging Probabilistic Reasoning and Crowdsourcing Techniques for Large-Scale Entity Linking. Proceedings of the 21st International Conference on World Wide Web (WWW '12). New York, NY, USA: ACM, 2012, pp. 469–478. doi: 10.1145/2187836.2187900

[4]. S. M. Yimam, I. Gurevych, R. E. de Castilho, C. Biemann. 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. Sofia, Bulgaria: Association for Computational Linguistics, 2013, pp. 1–6.

[5]. В. Бочаров, С. Алексеевич, Д. Грановский, Е. Протопопова, М. Степанова, А. Суриков. Морфологическая разметка корпуса силами волонтёров. Компьютерная лингвистика и интеллектуальные технологии: По материалам ежегодной Международной конференции «Диалог» (Бекасово, 29 мая — 2 июня 2013 г.), вып. 12(19), Т. 1. Москва: Изд-во РГГУ, 2013, С. 109–124.

[6]. 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. Gothenburg, Sweden: Association for Computational Linguistics, 2014, pp. 101–104.

[7]. S. Lee, S. Park, S. Park. A Quality Enhancement of Crowdsourcing based on Quality Evaluation and User-Level Task Assignment Framework. 2014 International Conference on Big Data and Smart Computing (BIGCOMP). IEEE, 2014, pp. 60–65. doi: 10.1109/BIGCOMP.2014.6741408

[8]. M.-C. Yuen, I. King, K.-S. Leung. TaskRec: A Task Recommendation Framework in Crowdsourcing Systems. Neural Processing Letters, pp. 1–16, 2014. doi: 10.1007/s11063-014-9343-z

[9]. D. R. Karger, S. Oh, D. Shah. Budget-Optimal Task Allocation for Reliable Crowdsourcing Systems. Operations Research, vol. 62, no. 1, pp. 1–24, 2014. doi: 10.1287/opre.2013.1235

[10]. P. Welinder P. Perona. Online crowdsourcing: Rating annotators and obtaining cost-effective labels. 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2010, pp. 25–32. doi: 10.1109/CVPRW.2010.5543189

[11]. D. E. Difallah, G. Demartini, P. Cudré-Mauroux. Pick-A-Crowd: Tell Me What You Like, and I'll Tell You What to Do. Proceedings of the 22Nd International Conference on World Wide Web (WWW '13). Rio de Janeiro, Brazil: International World Wide Web Conferences Steering Committee, 2013, pp. 367–374.

[12]. M. Daltayanni, L. de Alfaro, P. Papadimitriou. WorkerRank: Using Employer Implicit Judgements to Infer Worker Reputation. Proceedings of the Eighth ACM International Conference on Web Search and Data Mining (WSDM '15). New York, NY, USA: ACM, 2015, pp. 263–272. doi: 10.1145/2684822.2685286

[13]. A. Sheshadri, M. Lease. SQUARE: A Benchmark for Research on Computing Crowd Consensus. First AAAI Conference on Human Computation and Crowdsourcing, 2013, pp. 156–164.

[14]. C. M. Meyer, M. Mieskes, C. Stab, I. Gurevych. DKPro Agreement: An Open-Source Java Library for Measuring Inter-Rater Agreement. Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: System Demonstrations. Dublin, Ireland: Dublin City University and Association for Computational Linguistics, 2014, pp. 105–109.

[15]. B. Satzger, H. Psaier, D. Schall, S. Dustdar. Auction-based crowdsourcing supporting skill management. Information Systems, vol. 38, no. 4, pp. 547–560, 2013. doi: 10.1016/j.is.2012.09.003

[16]. Y. Gao, A. Parameswaran. Finish Them! : Pricing Algorithms for Human Computation. Proceedings of the VLDB Endowment, vol. 7, no. 14, 2014. doi: 10.14778/2733085.2733101

[17]. L. Tran-Thanh, T. D. Huynh, A. Rosenfeld, S. D. Ramchurn, N. R. Jennings. Crowdsourcing Complex Workflows under Budget Constraints. Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15). AAAI Press, 2015, pp. 1298–1304.

[18]. M. Hosseini, K. Phalp, J. Taylor, R. Ali. The Four Pillars of Crowdsourcing: a Reference Model. 2014 IEEE Eighth International Conference on Research Challenges in Information Science (RCIS), 2014, pp. 1–12. doi: 10.1109/RCIS.2014.6861072

[19]. А. Панченко, Н. В. Лукашевич, Д. Усталов, Д. Паперно, К. М. Мейер, Н. Константинова, RUSSE: семинар по оценке семантической близости для русского языка. Компьютерная лингвистика и интеллектуальные технологии: По материалам ежегодной Международной конференции «Диалог» (Москва, 27 — 30 мая 2015 г.). М.: Изд-во РГГУ, 2015, вып. 14(21), Т. 2, С. 89–105.