CrowdGame: A Game-Based Crowdsourcing System for Cost-Effective Data Labeling

Tongyu Liu
Renmin University of China
ltyzzz@ruc.edu.cn

Jingru Yang
Renmin University of China
hinsonver@ruc.edu.cn

Ju Fan*
Renmin University of China
fanj@ruc.edu.cn

Zhewei Wei
Renmin University of China
zhewei@ruc.edu.cn

Guoliang Li
Tsinghua University, China
liguoliang@tsinghua.edu.cn

Xiaoyong Du
Renmin University of China
duyong@ruc.edu.cn

ABSTRACT

Large-scale data labeling has become a major bottleneck for many applications, such as machine learning and data integration. This paper presents CrowdGame, a crowdsourcing system that harnesses the crowd to gather data labels in a cost-effective way. CrowdGame focuses on generating high-quality labeling rules to largely reduce the labeling cost while preserving quality. It first generates candidate rules, and then devises a game-based crowdsourcing approach to select rules with high coverage and accuracy. CrowdGame applies the generated rules for effective data labeling. We have implemented CrowdGame and provided a user-friendly interface for users to deploy their labeling applications. We will demonstrate CrowdGame in two representative data labeling scenarios, entity matching and relation extraction.

ACM Reference Format:
Tongyu Liu, Jingru Yang, Ju Fan, Zhewei Wei, Guoliang Li, and Xiaoyong Du. 2019. CrowdGame: A Game-Based Crowdsourcing System for Cost-Effective Data Labeling. In *2019 International Conference on Management of Data (SIGMOD '19), June 30-July 5, 2019, Amsterdam, Netherlands.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3299869.3320221

1 INTRODUCTION

In many real-world applications, such as machine learning and data integration, gathering a sufficient number of *labels* for datasets has become a bottleneck to effective data analytics. For example, as a major advancement in machine learning, deep learning usually requires massive training labels to achieve superior performance.

Crowdsourcing is recently utilized to solve the bottleneck by asking the crowd to label data at low price [2, 3, 5]. For example, the well-known ImageNet dataset [1] is constructed by gathering image labels from the crowd on Amazon Mechanical Turk (AMT). However, many real datasets contain tens of thousands to millions of data tuples to be labeled, thus incurring a challenge of high crowdsourcing cost.

To address the problem, we introduce CrowdGame, a game-based crowdsourcing system for *cost-effective* data labeling, with the following salient features.

Utilization of labeling rules. To reduce cost, CrowdGame utilizes *labeling rules* that can assign labels to a batch of tuples in the dataset. It applies both *user-defined* rules specified by end-users and *weak-supervision* rules automatically generated by algorithms. However, the rules may have diverse quality on *coverage* and *accuracy*. Thus, CrowdGame focuses on generating the rules that not only cover a large proportion of the dataset, but also provide reliable labels.

High-quality rule generation via crowdsourcing. We develop a *game*-based crowdsourcing strategy to generate high-quality labeling rules. The strategy employs two groups of crowd workers: one group directly validates the rules to play a role of *rule generator*, while the other checks tuples covered by the rules as a *rule refuter*. We let the two groups play an adversarial game: rule generator identifies rules with significant improvement on coverage, while rule refuter tries to refute rule generator by checking representative tuples that degrade accuracy of the rules. We introduce a minimax strategy and develop efficient task selection algorithms.

Rule-based labeling model. CrowdGame applies the generated rules for effective data labeling by developing rule pruning and quality-aware rule aggregation techniques.

Equipped with these features, CrowdGame can significantly reduce crowdsourcing cost, e.g., by one order of magnitude in applications like entity matching, while still preserving high quality on labeling results. This is powered by

*Ju Fan is the corresponding author.

SIGMOD '19, June 30-July 5, 2019, Amsterdam, Netherlands

ACM ISBN 978-1-4503-5643-5/19/06…$15.00
https://doi.org/10.1145/3299869.3320221

our techniques proposed in [7]. Note that there are some related works, like Snorkel [4, 6], that also utilize labeling rules (or functions) for data labeling. Nevertheless, the difference is that CrowdGame develops a *crowdsourcing* method to select high-quality rules from very noisy candidates, while Snorkel focuses on "consolidating" the rules.

Demonstration Scenarios. We demonstrate CrowdGame in two representative data labeling scenarios. (1) *Entity Matching*. We will show how CrowdGame is used to label whether product records from different sources represent the same entity. We will allow the participants to upload an unlabeled dataset and demonstrate how CrowdGame generates labeling rules and visualizes the game-based crowdsourcing process. (2) *Relation Extraction*. We will demonstrate CrowdGame in extracting the spouse relation between two person entities from a sentence. We will show how CrowdGame generates labeling rules by identifying textual patterns, and leverages game-based crowdsourcing to obtain high-quality rules for effective labeling.

2 SYSTEM OVERVIEW

The architecture of the CrowdGame system is shown in Figure 1. CrowdGame takes as input an unlabeled dataset of a set of *tuples*, and aims to assign *labels* to the tuples. CrowdGame leverages the crowd (aka. workers) on crowdsourcing platforms, such as AMT. To reduce crowdsourcing cost, it utilizes *labeling rules* (or rules for simplicity).

Labeling Rules. A labeling rule is a user-defined or automatically generated heuristic that assigns a label to a subset of tuples in the dataset. However, it is very challenging to generate *high-quality* rules that not only cover a large proportion of the dataset, but also provide reliable labels.

Game-Based Crowdsourcing for Rule Generation. To generate high-quality rules, CrowdGame first generates a set of user-defined or weak-supervision rules as candidates (Section 2.1). As some candidates may have low quality, it then solicits the crowd to identify "good" rules from noisy candidates. Section 2.2 presents a game-based task selection technique to achieve this goal. Finally, CrowdGame uses a rule-based labeling model for effective data labeling (Section 2.3). Note that this paper only provides high-level idea of the techniques due to the space limit. Refer to our paper [7] for details on algorithms and empirical evaluation.

User Interface. CrowdGame provides a user-friendly interface to allow the users to interact with the process of data labeling. It visualizes the dynamic changes of the labeling dataset based on the crowdsourcing answers collected so far. It allows the end-user to interactively refine labeling rules based on his/her domain knowledge or the observation from current labeling dynamics. Moreover, it enables the user to easily control the labeling progress.

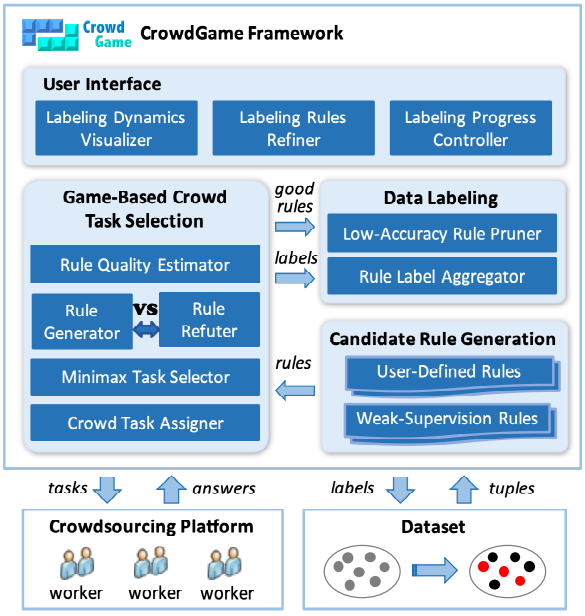


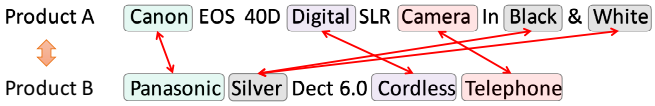
Figure 1: Architecture of the CrowdGame system.

2.1 Candidate Rule Generation

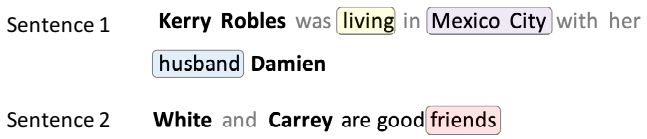
To generate candidate rules, CrowdGame uses two widely-used methods. *User-defined* rules ask users of CrowdGame to write labeling heuristics based on their domain knowledge. *Weak supervision* rules are recently extensively studied [4, 6] to assign (possibly) noisy labels to unlabeled data. CrowdGame allows users to customize algorithms for generating weak-supervision rules tailored for different labeling scenarios. For example, it provides the following algorithms for entity matching and relation extraction respectively.

Blocking rules for entity matching: Consider entity matching that labels if two products, as shown in Figure 2(a), represent the same entity (label 1) or not (label −1). CrowdGame generates *blocking rules* that assign label −1 to record pairs. As there are limited works on blocking rules on textual data, we propose to identify *discriminative keyword pairs* to discriminate record pairs. For example, ⟨`Canon`, `Panasonic`⟩ tends to be discriminative, because one record with `Canon` is unlikely to match another one with `Panasonic`. In contrast, ⟨`Digital`, `Camera`⟩ may not be discriminative. CrowdGame applies an effective algorithm to identify keyword pairs by using word embedding and word mover's distance [7].

Pattern-based rules for relation extraction: CrowdGame uses textual patterns as rules for extracting `spouse` relation. As shown in Figure 2(b), "`husband`" occurring close to *Kerry Robles* and *Damien* may be good at identifying `spouse` relation for these two entities (labeling 1), while "`friends`" can be taken as a *negative* rule that assigns label −1. CrowdGame develops distant supervision and phrase detection to generate candidate patterns.



(a) Blocking rules for entity matching.



(b) Pattern-based rules for relation extraction.

Figure 2: Examples of candidate rules generation.

2.2 Game-Based Crowd Task Selection

CrowdGame develops the following crowdsourcing techniques to select "high-quality" rules from candidates.

Formalization of rule quality. Intuitively, we aim to identify a set of "high-quality" rules with two objectives: (1) *high coverage*: selecting the rules that together cover as many tuples as possible, so as to ensure a large proportion of data receives labels; (2) *high accuracy*: preferring the rules that induce few wrong labels. Naturally, there may be tradeoff between coverage and accuracy. A fine-tuned rule set may be very accurate, but has limited coverage, while rules with significant coverage may produce many wrong labels. We define the *loss* of a rule set as a weighted combination of the number of uncovered tuples and the number of mislabeled tuples. The lower the loss of a rule set, the higher the quality of the rule set. Thus, we can balance the tradeoff between accuracy and coverage by minimizing the *loss* of rule set.

Two pronged crowd task scheme. CrowdGame considers two types of crowdsourcing tasks to select good rules. The first one is *rule validation* task that asks the crowd to check whether a rule is accurate or not. However, without inspecting specific tuples, the crowd sometimes gives low-quality answers for a rule validation task, as they may not know if the rule works for the current dataset. Thus, CrowdGame also utilizes *tuple checking* task that employs the crowd to give the label of a tuple and uses the result to validate/invalidate the rules covering the tuple. However, it is expensive to crowdsource many tuple checking tasks.

Game-based crowd task selection. To reduce crowdsourcing cost, CrowdGame introduces a *game-based* task selection strategy that formalizes task selection as a two-player adversarial game with our *rule set loss* as the game penalty. One player is Rule Generator that employs the crowd to answer rule validation tasks on some selected rules to minimize the loss. The other player is Rule Refuter that tries to refute the rule generator. It asks the crowd to check some representative tuples that have large chance to "degrade" the accuracy of the crowd-validated rules, so as to maximize the loss. To support this, we develop a *minimax* task selection algorithm: It iteratively calls Rule Generator to minimize the loss and Rule Refuter to maximize the loss until crowdsourcing budget is used up. Our empirical evaluation shows CrowdGame can significantly reduce crowdsourcing cost while still preserving high labeling quality [7].

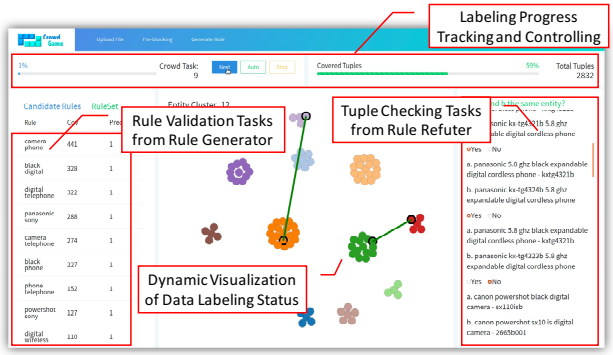


Figure 3: User Interface of CrowdGame.

2.3 Data Labeling with Rules

CrowdGame selects the crowd-validated rules from the candidates, and refines the rules by pruning low-accuracy ones. We develop a *quality-aware label aggregation* method that applies voting mechanisms while considering quality of rules. For the uncovered tuples, CrowdGame can use tuple checking tasks to obtain their labels, or infer the labels by applying a machine-learning model trained on the labeled dataset.

3 DEMONSTRATION SCENARIOS

We will present a web app to the participants, and illustrate CrowdGame in two data labeling applications, entity matching and relation extraction. A demonstration video can be found in Youtube[1]. Figure 3 shows the user interface of CrowdGame. The window in the middle visualizes the dynamic changes of data labeling. The left and right windows respectively show rule validation and tuple checking tasks selected by Rule Generator and Rule Refuter.

Entity Matching. We demonstrate CrowdGame in labeling whether records from different e-commerce websites represent the same product. Specifically, we allow the participants to upload a CSV file containing the records with attributes, `id`, `name` (e.g., *Canon PowerShot Silver Digital Camera - SX110IS*) and `source` (either Abt or BestBuy). Then, CrowdGame utilizes crowdsourcing to label *pairs* of records as tuples.

Scenario I - Visualizing Data Labeling Status. CrowdGame employs a graphical way to visualize current labeling status, as shown in Figure 4. Each node represents a record, and an edge represents a record pair to be labeled. The nodes grouped together in clusters represent different *blocks*

[1]https://youtu.be/0F9QvSW_dUM

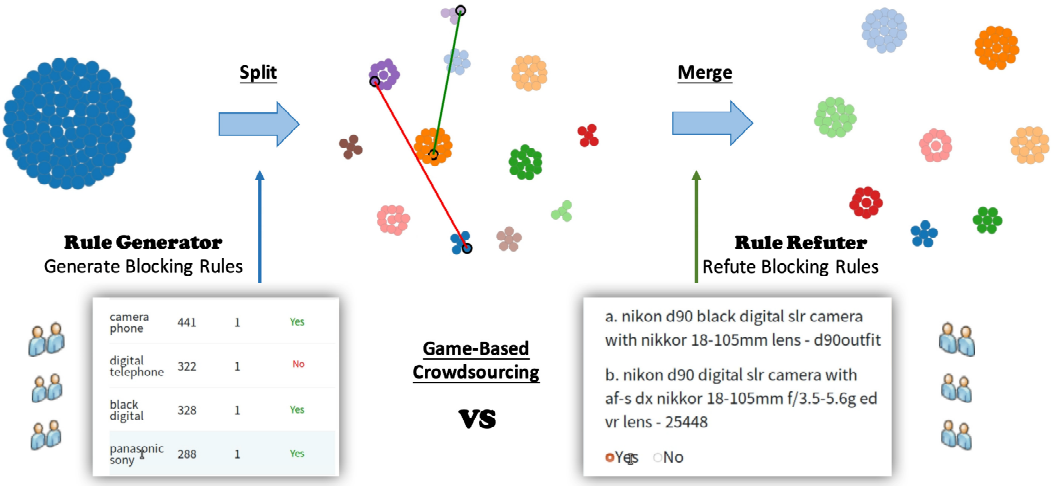


Figure 4: Demonstration scenarios of CrowdGame for entity matching.

where only records within same blocks are considered to be matched as same entities. Thus, the labeling process can be visually interpreted as "splitting" all nodes into many small clusters, each of which represents a distinct entity.

Scenario II - Generating Candidate Blocking Rules. CrowdGame offers a built-in algorithm to identify *discriminative keyword pairs* as candidate blocking rules, e.g., ⟨Panasonic, Sony⟩ in Figure 4. Utilizing the rules visually "split" current clusters into small ones, since records covered by them should reside in different clusters.

Scenario III - Visualizing Dynamic Changes for Game-Based Crowdsourcing. We allow the participants to run game-based crowdsourcing by either on-the-fly collecting crowd answers, or using our pre-fetched crowd answers. CrowdGame will show how Rule Generator and Rule Refuter affect the labeling status. The crowd validates/invalidates the rules selected by Rule Generator. For example, as shown in Figure 4, after applying the crowd-validated rules, the records are split into 12 clusters. Then, Rule Refuter selects pairs of records from different clusters (illustrated by the lines between nodes) with high chance to be matched, e.g., the `Nikon D90` example. After checked by the crowd, these matched record pairs can be used to degrade the selected rules and thus merge small clusters into bigger ones. In such a way, CrowdGame visualizes the dynamic changes of game-based crowdsourcing by splitting and merging record clusters. If the clusters tend to be stable without significant splitting and merging, the participants can terminate game-based crowdsourcing for saving cost.

Relation Extraction. We demonstrate CrowdGame in extracting the `spouse` relation from text. We allow the participants to upload a `CSV` file where each tuple contains two person names and a sentence containing the names. We show how CrowdGame assigns labels for `spouse` relation.

Note that CrowdGame can be easily applied to other data labeling tasks. The key of the application is to devise task-specific methods for generating candidate rules, such as Sherlock rules for data cleaning, transformation rules for schema matching, etc.

4 CONCLUSION

In this paper, we introduced our CrowdGame system for cost-effective data labeling. We presented an overview of system implementation and demonstrated its salient features on user-friendly interface, game-based crowdsourcing for labeling rule generation, and labeling performance.

Acknowledgment. This work was supported by the 973 Program of China (2015CB358700), NSFC (61632016, 61602488, U1711261, 61472198, 61521002, 61661166012, 61502503), the Research Funds of Renmin University of China (18XNLG18, 18XNLG21), and the Humanities and Social Sciences Base Foundation of MOE of China (16JJD860008).

REFERENCES

[1] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and F. Li. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009.
[2] J. Fan, G. Li, B. C. Ooi, K. Tan, and J. Feng. icrowd: An adaptive crowdsourcing framework. In *SIGMOD*, pages 1015–1030, 2015.
[3] D. Gao, Y. Tong, J. She, T. Song, L. Chen, and K. Xu. Top-k team recommendation and its variants in spatial crowdsourcing. *Data Science and Engineering*, 2(2):136–150, 2017.
[4] A. Ratner, S. H. Bach, H. R. Ehrenberg, J. A. Fries, S. Wu, and C. Ré. Snorkel: Rapid training data creation with weak supervision. *PVLDB*, 11(3):269–282, 2017.
[5] Y. Tong, L. Chen, Z. Zhou, H. V. Jagadish, L. Shou, and W. Lv. SLADE: A smart large-scale task decomposer in crowdsourcing. *IEEE Trans. Knowl. Data Eng.*, 30(8):1588–1601, 2018.
[6] P. Varma and C. Ré. Snuba: Automating weak supervision to label training data. *PVLDB*, 12(3):223–236, 2018.
[7] J. Yang, J. Fan, Z. Wei, G. Li, T. Liu, and X. Du. Cost-effective data annotation using game-based crowdsourcing. *PVLDB*, 12(1):57–70, 2018.