Distrib Parallel Databases (2015) 33:95–122
DOI 10.1007/s10619-014-7163-9

Crowdsourcing large scale wrapper inference

Valter Crescenzi · Paolo Merialdo · Disheng Qiu

Published online: 29 October 2014

Abstract We present a crowdsourcing system for large-scale production of accurate wrappers to extract data from data-intensive websites. Our approach is based on supervised wrapper inference algorithms which demand the burden of generating training data to workers recruited on a crowdsourcing platform. Workers are paid for answering simple queries carefully chosen by the system. We present two algorithms: a single worker algorithm (ALF_η) and a multiple workers algorithm (ALFRED). Both the algorithms deal with the inherent uncertainty of the workers' responses and use an active learning approach to select the most informative queries. ALFRED estimates the workers' error rate to decide at runtime how many workers should be recruited to achieve a quality target. The system has been fully implemented and tested: the experimental evaluation conducted with both synthetic workers and real workers recruited on a crowdsourcing platform show that our approach is able to produce accurate wrappers at a low cost, even in presence of workers with a significant error rate.

Keywords Data extraction · Wrapper induction · Crowdsourcing

1 Introduction

The abundance of data contained in web pages has motivated many research efforts towards the development of effective methods and tools for generating web wrappers

V. Crescenzi · P. Merialdo (✉) · D. Qiu
Dipartimento di Ingegneria, Università degli Studi Roma Tre, Via della Vasca Navale, 79, 00146 Rome, Italy
e-mail: merialdo@dia.uniroma3

V. Crescenzi
e-mail: crescenz@dia.uniroma3.it

D. Qiu
e-mail: disheng@dia.uniroma3.it

to extract information from data intensive websites, i.e., websites that publish large amounts of data in pages generated by scripts that embed into HTML templates data from a back-end database.

To alleviate the burden of writing wrappers manually, several inference approaches have been investigated. Supervised approaches (e.g., [16,24]) infer the wrapper from a set of training data, typically provided as labeled values. Although they can produce accurate solutions, they have limited scalability, mainly because of the need of a human intervention to produce the labeled values. Unsupervised approaches (e.g., [3, 9]) have been proposed as an attempt to "scale-up" the wrapper generation process by overcoming the need of labeled values. Unfortunately, they have limited applicability because the quality of the produced wrappers is highly unpredictable, and the inference process can be hardly controlled towards better solutions.

Crowdsourcing platforms represent an intriguing opportunity to "scale-out" supervised wrapper inference approaches. These platforms support the assignment of mini-tasks to human workers recruited on the Web: since they can involve large numbers of workers, they allow the production of massive amounts of labeled values, which can support wrapper inference at large scale.

However, inferring wrappers with the support of workers recruited on crowdsourcing platforms introduces a number of challenges that were not addressed in the literature:

- the tasks submitted to the platform should be extremely simple, since they are performed by non-expert workers;
- the number of tasks submitted to the crowd should be minimized to reduce the wrappers production costs;
- the wrapper inference algorithm should be resilient to labeling errors, since workers recruited on crowdsourcing platforms might be inaccurate.

This paper presents ALFRED, a system that addresses the above issues by engaging crowd workers to supervise the wrapper generation process. ALFRED progressively infers a wrapper by posing simple questions, *membership queries* (*MQ*), on the contents of a web page. Because of their simplicity, membership queries are suitable for unskilled workers recruited on crowdsourcing platforms: as they admit only a yes/no answer, they represent the simplest form of questions to support a learning process [2]. The answers to the queries posed by the system represent labeled values, which are then used to feed the learning process.

ALFRED builds on ALF$_\eta$, a supervised wrapper induction algorithm that adopts consolidated active learning techniques [34] to select the queries that more quickly bring to the generation of an accurate wrapper. ALF$_\eta$ relies on a probabilistic quality model that considers the presence of errors in the answers returned by the workers. ALFRED (ALF$_\eta$ with REDundancy) improves the resilience to noisy answers by recruiting several workers on the same task. In particular, ALFRED schedules crowdsourcing tasks composed of queries to infer wrappers for several attributes, among which a subset are *redundant*, i.e., assigned to multiple workers.

ALFRED exploits the agreement among multiple workers to jointly estimate their error rates and the quality of the wrappers inferred: the larger the amount of redundancy,

the better the estimation of the workers error rates and of the wrapper quality, but the fewer resources are left for the inference process.

Our experimental evaluation shows that during the learning process, ALFRED adaptively schedules the amount of redundant attributes that should be allocated in each task. For the twofold purpose of minimizing the costs and achieving the target quality, it engages additional workers for an attribute only when it is necessary to compensate the noise introduced by the workers.

1.1 Contribution

To the best of our knowledge, ALFRED represents the first proposal that exploits crowdsourcing to address wrapper inference. The paper makes several contributions: (i) we introduce an algorithm that exploits consolidated active learning techniques [34] and a bayesian model to infer a wrapper from the noisy labeled values provided by workers recruited on a crowdsourcing platform; (ii) we develop a solution to decide at runtime how many workers should be recruited to deal with the presence of noisy answers; (iii) we present techniques to leverage redundant tasks to estimate the workers' error rates during the learning process; (iv) we report the results of an extensive experimental activity conducted with both synthetic and real workers recruited from a crowdsourcing platform.

This paper is an extended version of previous conference and workshop papers [7,8]: in [7] we studied wrapper inference with a single and infallible worker, while in [8] we introduced a preliminary development of the inference algorithm in the form presented in this paper, considering erroneous answers and the possibility of recruiting multiple workers. The material in Sect. 3 that describes how we generate the extraction rules and Sect. 6, which investigates how to schedule the tasks to submit to a crowdsourcing platform, is new. We also provide substantially increased coverage and depth of the experimental activities reported in Sect. 7, as well as a larger discussion of related work in Sect. 8.

1.2 Paper outline

The paper is organized as follows: Sect. 2 introduces some preliminary notions to define the problem, and presents an overview of our solution, which is then developed in the successive sections. In particular, Sect. 3 describes how we set up the inference process, generating a set of candidate wrappers and sampling the input pages. Section 4 presents ALF_η, the active learning algorithm for selecting the correct wrapper with the support of a worker recruited on a crowdsourcing platform. Section 5 discusses how we estimate the error rate of the workers introducing redundant tasks. Section 6 presents ALFRED, an algorithm to schedule the tasks to submit to the crowdsourcing platform. Section 7 briefly describes the implementation of a system protype, and then reports the experimental results. Section 8 discusses related work, and Sect. 9 concludes the paper.

2 The ALFRED approach

We focus on data extraction from data-intensive websites whose pages are generated by scripts that embed data from an underlying database into an HTML template. Let $U = \{p_1, \ldots, p_n\}$ be an ordered set of pages generated by the same script. Given an attribute A of interest published in the pages, its values can be extracted by means of an *extraction rule* (or simply *rule*). The value extracted by a rule r from a page p, denoted by $r(p)$, is either a string occurrence from the HTML source code of p, or a special nil value to denote that the rule cannot extract any value from the page. A rule r applied over the pages in U returns a vector, i.e., an ordered set of values $r(p_1), \ldots, r(p_n)$, denoted by $r(U)$, indexed by the pages of U.

We assume the existence of an initial annotated page, $p_1 \in U$, i.e., one page where the value v_A of the target attribute A has been marked. The input annotated page may be supplied either manually, by a user that marks the string occurrence of the value, or automatically, by looking up in the page values from an available database of known values for A.

Given such an initial annotated page p_1, we generate a set $\mathcal{R}_A$ of rules, all extracting v_A from p_1, i.e., such that $\forall r \in \mathcal{R}_A, r(p_1) = v_A$.

Since the pages of U are produced by the same generating script according to a shared HTML template, ideally the rules in $\mathcal{R}_A$ should behave identically on every page. However, web pages usually exhibit variations that prevent these rules to extract the correct values over the whole set of pages U. For example, to accomodate for the presence of an optional attribute, scripts can introduce mutually exclusive HTML fragments corresponding either to the presence or to the absence of its value. Similarly, advertisements, or other special contents can introduce local variations in the generated pages. Therefore, despite the regularities in the structure of the HTML code over pages from data intensive websites, rules that correctly extract an attribute value from one sample page might not work correctly for other pages in U, as illustrated in the following example.

Example 1 Suppose that we are interested to extract the Title from the set of fictional movie pages $U = \{p_1, p_2, p_3\}$ whose DOM trees are sketched in Fig. 1. Assume that the initial annotation $v_{\text{Title}} =$ 'City of God' is supplied on the sample page p_1. Figure 2 shows a set $\mathcal{R}_{\text{Title}} = \{r_1, r_2, r_3\}$ of candidate rules generated from this initial annotation. Note that even if all rules extract the initial annotation from p_1, r_1 is the only one that correctly extracts the movie title from the other pages, whereas r_2 does

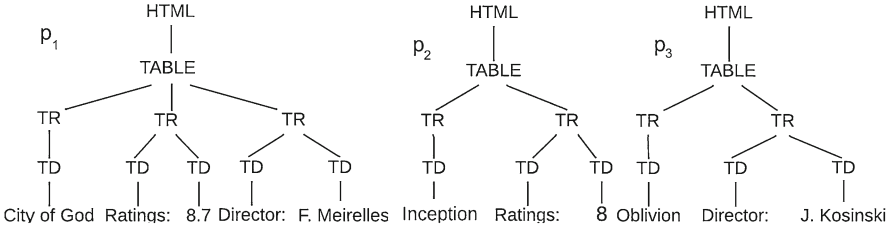

Fig. 1 DOM of three sample pages

Fig. 2 Extraction rules in $\mathcal{R}_{\mathsf{Title}} = \{r_1, r_2, r_3\}$ with initial annotation '**City of God**' and the corresponding set of values V_{Title} that they extract from pages in $U = \{p_1, p_2, p_3\}$ shown in Fig. 1

rules \ pages		U		
		p_1	p_2	p_3
$\mathcal{R}_{\mathsf{Title}}$	r_1	City of God	Inception	Oblivion
	r_2	City of God	Inception	nil
	r_3	City of God	nil	Oblivion

r_1 =/html/table/tr[1]/td
r_2 =//td[contains(.,"Rating:")]/../../tr[1]/td
r_3 =//td[contains(.,"Director:")]/../../tr[1]/td

not work on p_3 (the page of a movie without user ratings), and r_3 does not work on p_2 (a movie without a director).

We say that an extraction rule r is *correct* for an attribute A if r extracts from every page $p \in U$ either the string occurrence of the value of A published in p, or nil if p does not publish any value for A. With the notion of correct extraction rule we can state our problem as follows: given an input collection of pages U, and an annotation for the attribute A in a page $p_1 \in U$, find a *correct* extraction rule for attribute A.

2.1 Solution overview

We propose an algorithm, ALF$_\eta$, that progressively infers the correct rule by posing simple questions to human workers recruited on a crowdsourcing platform.

ALF$_\eta$ evaluates the candidate rules by posing a sequence of *membership queries:* ALF$_\eta$ shows a web page and asks the worker whether a given value present in the page corresponds to the correct value for a target attribute (e.g., "*Is the string* `'Inception'` *the title of the movie in this page?")*. The binary answer (yes/no) provided by the worker feeds a *training sequence*, which is used by a Bayesian model to compute the probability of correctness of the candidate rules. As new queries are posed and their answers are collected, the training sequence is expanded and the probabilities are updated, until a termination condition is satisfied.

To reduce the number of queries, we exploit active learning techniques: our algorithm poses its queries by choosing the value, among those extracted by the candidate rules, about which it is least confident, according to the probability of correctness gained with the previous answers. Our inference algorithm does not run the inference task on the whole set of pages, but on a small yet representative sample. In a preprocessing phase a simple yet principled and effective sampling algorithm, PAGESAMPLER, selects the set of sample pages over which the candidate rules exhibit all their differences.

Our probabilistic model also considers that the worker can respond with wrong answers. The performances of the algorithm and of the underlying probabilistic model are strongly affected by a correct estimation of the worker's error rate: if the error-rate is overestimated, the algorithm does not trust the worker and ends up posing more questions than actually needed, thus raising the costs. Conversely, if the error rate is underestimated, it poses less queries, eventually compromising the quality of the results. To overcome this issue, we submit the same task to multiple workers and

extend our approach in order to estimate a worker error rate based on the agreement with other workers.

Such an approach raises the question of how much redundancy is actually needed to reach an optimal trade-off between costs and quality. Our solution to this problem consists of an involved algorithm, ALFRED, that adaptively schedules the tasks submitted to the crowdsourcing platform in order to decide at runtime the degree of redundancy needed to obtain reliable results at low costs. ALFRED submits tasks composed by several attributes with a partial overlapping: each task contains attributes which are included also in another task. With the results of the redundant portion of each task, ALFRED estimates the error rate of the worker that accomplished the task, and uses such error rate estimation to compute the probability of correctness of the extraction rules for the remaining attributes of the task.

3 Generation of the candidate extraction rules

In this section we describe how we generate a set of candidate rules. We consider extraction rules belonging to a simple fragment of XPath. Given the initial annotated page, we compute several rules that specify paths starting from a *pivot* node and leading to the annotated value.

The process to generate candidate extraction rules is articulated in two steps. First, we locate nodes that are likely to be part of underlying template. Following the intuition developed in [3], we classify as part of the template those nodes that occur exactly once with same root-to-node path (without indices) in a significant percentage (40 %) of the input pages.[1] The rationale is that these properties do not hold by chance; rather, they are a consequence of the presence of the underlying HTML template used by the script to create the pages. Notice that according to these heuristics, the document root <html> and all the tags having an 'id' attribute are classified as template node since they occur exactly once in every page.

In the second step, extraction rules are generated as XPath expressions specifying a path starting from a template node P, which we call the *pivot* of the rule, to the leaf node V containing the annotated value v. This expression is obtained by appending three components: (i) an expression that matches at most one pivot node P (examples of these expressions are: /html, //table or //td[@id='key']); (ii) the path from P to the first ancestor node, N, shared by P and V: if P is not an ancestor of V, these expressions are in the form ../../ … to follow the parent-axis, otherwise it is just the empty string; (iii) a path descending from the common ancestor node N and reaching *only* the target textual node V. Besides the complete sequence of parent to child nodes, several alternative expressions are generated, each including the presence of the // operator at different depths in the path (examples of these expressions are /tr[1]/td, /tr[1]//text(), //text()).

Example 2 Figure 2 shows (a subset of) the extraction rules $\mathcal{R}_{\text{Title}} = \{r_1, r_2, r_3\}$ generated from the initial annotation 'City of God' on the sample page p_1, whose DOM tree is shown in Fig. 1.

[1] Actually, we do not consider the whole set of pages U, but a much smaller sample set including the initial annotated page and at most 100 pages randomly chosen from U.

Assume $U = \{p_1, p_2, p_3\}$. Since nodes <html> and <table> appear exactly once in every page, they are classified as template nodes, as well as nodes 'Directors:' and 'Ratings:', which occur in two out of three pages. All the other nodes do not fall in this category, because they either occur once in less than 40 % of pages (e.g., '8.7' occurs in one page out of three), or they occur more than once for page (e.g., <tr> and <td>).

Rule r_1 is pivoted in the document root, rule r_2 and r_3 are pivoted in the template nodes 'Rating:' and 'Director:', respectively. (Other candidate rules, such as //table/tr[1]/td pivoted on <table> are not reported for the sake of space).

To avoid an excessive proliferation of rules, we bound the length of the extraction rules, i.e., the number of XPath steps composing its expression. Namely, we have empirically observed that producing extraction rules longer than 8 steps does not produce any benefit.

3.1 Representative sample set

As we discussed in Sect. 2, although the candidate rules are correct for the page containing the initial annotation, they might not work correctly for other pages in U, because of local variations on the HTML structure of the pages. After the candidate rules have been generated, we perform a procedure for computing a small set of sample pages that manifest all the variations, observable by the candidate rules, that may occur in the whole, potentially very large, set of pages U.

Our goal is to find a set $I \subseteq U$ such that $|I| \ll |U|$ yet I is *representative* with respect to a given set of candidate extraction rules $\mathcal{R}_A$ of all the pages in U.

In our context, the concept of representativeness of a set of pages $I \subset U$ with respect to a set of rules $\mathcal{R}_A$ can be formalized by introducing the *disagreement set* of two extraction rules.

Given a set of pages P, and a set of rules $\mathcal{R}_A$, the disagreement set, denoted as $D^P(r_i, r_j)$, between two rules $r_i, r_j \in \mathcal{R}_A$, is the set of pages in P making observable their differences: $D^P(r_i, r_j) = \{p \in P : r_i(p) \neq r_j(p)\}$, i.e., the subset of pages in P on which r_i and r_j extract different values. Two rules r_i, r_j extract from P the same values, and hence are indistinguishable for our purposes, if and only if $D^P(r_i, r_j) = \emptyset$.

We say that a subset $I \subseteq U$ is *representative* of U with respect to a set of rules $\mathcal{R}_A$ if and only if:

$$\forall r_i, r_j \in \mathcal{R}_A, [D^I(r_i, r_j) = \emptyset \iff D^U(r_i, r_j) = \emptyset].$$

In other terms, I is representative of U with respect to $\mathcal{R}_A$ if all the differences amongst the rules in $\mathcal{R}_A$ are also observable on I.

Example 3 Consider again our running example in Fig. 2. Let us assume $U = \{p_1, p_1, p_3\}$. The sample $I = \{p_1, p_2\}$ is not representative with respect to $\mathcal{R}_A = \{r_1, r_2, r_3\}$ since $D^U(r_1, r_3) = \{p_2\}$ whereas $D^I(r_1, r_2) = \emptyset$.

Given the set of input pages U, there exist many representative subsets, including U itself. As discussed above, our goal is to find a small sample set. Finding the smallest

one is an instance of the well-known SET COVER problem: a page differentiates the set of rules that extract distinct values from it.[2] Actually we do not need to compute the optimal sample set: it suffices to approximate it by considering a small but not necessary minimal set of pages: in any real setting the additional pages included in the sample set do not make any practical difference.

Listing 1 introduces PAGESAMPLER, a greedy sampling algorithm to extract a representative set of pages I w.r.t. a set of rules $\mathcal{R}_A$ from a set of input pages U in $O(|U| \cdot |\mathcal{R}_A|)$ time and $O(|\mathcal{R}_A|)$ space.

Listing 1 PAGESAMPLER: Sampling Algorithm

Input: a set of pages U;
Input: a set of rules R;

Output: a set $I \subseteq U$ that is *representative* of U w.r.t. R

```
1: let I = ∅;
2: let n = 0;
3: for p ∈ U do
4:   if (|R(I ∪ {p})| > n) then
5:     I ← I ∪ {p};
6:     n ← |R(I)|;
7:   end if
8: end for
9: return I;
```

PAGESAMPLER processes the set of pages U (lines 3–8). It maintains a set of pages I, initially empty, that is representative of the subset of pages already processed. It selects as representative only those pages that increase the number of different vectors extracted by the set of rules $\mathcal{R}_A$ (line 4). The pages selected according to this criterion make observable *new* differences between at least two rules that were otherwise indistinguishable in the subset of pages processed until the previous iteration.

Once the representative sample set is obtained, the set of candidate rules is organized in classes of equivalence. Distinct rules that produce the same results over the pages of the representative set are equivalent and are grouped in the same class. In the set of candidate rules $\mathcal{R}_A$, we save only one rule from every class of equivalent rules.[3]

In the next section, where we illustrate our solution to efficiently crowdsource the selection of the correct extraction rule from $\mathcal{R}_A$, we blur the distinction between the whole set of pages U and the set of representative pages I computed from U.

[2] The problem reduces to finding the smallest set of pages such that the union of the sets of rules differentiated from them equals the set of rules differentiated directly by U.

[3] We choose the rule with the shortest path, but other strategies, such as those discussed in [33] could be applied.

4 Crowdsourcing wrapper inference

Our inference process evaluates the candidate rules by posing a sequence of queries to a human *worker* recruited from a crowdsourcing platform. The worker is shown a page p and is asked whether a given value $v_A = r(p), r \in \mathcal{R}_A$, extracted from the candidate rule r is the correct one for the target attribute A in the page p. The binary answer l, with $l \in \{-, +\}$, provided by the worker adorns the queried value v_A with either a positive or a negative label, producing a *labeled value*, denoted by v_A^l.

Example 4 Continuing Example 1, the inference process can build a query with the value $r_1(p_2) = r_2(p_2) =$ 'Inception': the worker is shown page p_2, and is asked to confirm whether 'Inception' is the Title of the movie in that page. A confirmation corresponds to produce the labeled value Inception$^+$.

The labeled value returned by the worker is appended into a *training sequence* (t.s.), denoted L. A Bayesian model is used to compute the probability of correctness of the candidate rules, given the labeled value just acquired, and the t.s. collected so far. The process iterates, posing new queries and thus expanding the t.s., until a termination condition is satisfied.

Listing 2 reports the pseudo-code of the ALF$_\eta$ algorithm, which implements the inference approach. ALF$_\eta$ takes as input a set of pages U and a set of candidate rules $\mathcal{R}_A$ for the attribute A (computed from an initial annotated page); it poses queries to the worker and collects its answers into a t.s. L, and it returns a probability distribution function (p.d.f.) describing the probability of correctness of the rules in $\mathcal{R}_A$.

ALF$_\eta$ progressively builds a t.s. L by posing queries to a worker. In every iteration (lines 3–8), the worker is asked to label a new value v_A (lines 4–5). Then, for all the rules $r \in \mathcal{R}_A$, ALF$_\eta$ computes the probability distribution function $P(r|v_A, L)$, that is, the probability that each $r \in \mathcal{R}_A$ is correct, given the last labeled value and the t.s. L acquired so far (line 6). Finally, the t.s. L is expanded by adding v_A^l (line 7).

We now illustrate subprograms CHOOSEQUERY(), which implements an active learning strategy to choose the best value to be queried, and GOAL(), which implements the stopping condition for the algorithm. Then, in Sect. 4.1, we present the probabilistic model.

CHOOSEQUERY() is in charge of building the next query, that is, it selects the next value to be labeled by the worker. The value is picked up from the *set of candidate values* for attribute A, denoted V_A, which contains all the values extracted from pages in U by the candidate rules in $\mathcal{R}_A$: $V_A = \{r(p), r \in \mathcal{R}_A, p \in U\}$. Our strategy consists in choosing the value on which rules most disagree, appropriately weighted according to their probability. This is equivalent to compute the *vote entropy* for each $v \in V_A$:

$$H(v) = -[P(v^+|L) \log P(v^+|L) + P(v^-|L) \log P(v^-|L)]$$

where: $$P(v^+|L) = \sum_{r \in \mathcal{R}_A^v} P(r|L)$$

Listing 2 ALF$_\eta$: Active Learning Algorithm for Wrapper Inference

Input: a set of pages U
Input: the set of candidate rules $\mathcal{R}_A$

Parameter η: worker error rate
Parameter λ_r: target probability of correctness
Parameter λ_{MQ}: maximum budget

Ensure: a p.d.f. describing the probability of correctness of the rules in $\mathcal{R}_A$

1: **let** $L = \emptyset$;
2: $w \leftarrow$ ENGAGEWORKER();
3: **while** (**not** GOAL$(L, \lambda_r, \lambda_{MQ})$) **do**
4: $v_A \leftarrow$ CHOOSEQUERY(L);
5: $l \leftarrow w$.GETANSWER(v_A);
6: compute $P(r|v_A, L),\ \forall r \in \mathcal{R}_A$ with Eq. 1 and Eq. 2;
7: $L \leftarrow L \cup \{v_A^l\}$;
8: **end while**
9: **return** $P(r|L),\ \forall r \in \mathcal{R}_A$;

and $$P(v^-|L) = \sum_{r \in \mathcal{R}_A \setminus \mathcal{R}_A^v} P(r|L)$$

are the probabilities that v is respectively either a value to extract or an incorrect value: $\mathcal{R}_A^v$ is the set composed of rules in $\mathcal{R}_A$ that extract v.

Intuitively, the entropy measures the uncertainty of a value and querying the value with the highest entropy removes the most uncertain value:

$$\text{CHOOSEQUERY}(L)\{\textbf{return } \text{argmax}_{v \in V_A} H(v);\}$$

It is worth saying that other active learning strategies, such as *least confident* and *smallest margin*, on binary classification tasks as ours are equivalent to the above maximum entropy policy [34].

The most appropriate termination policy might well depend both on budget constraints and on the quality targets. We adopt a simple implementation of GOAL() that takes into account both aspects:

$$\text{GOAL}(L, \lambda_r, \lambda_{MQ})\ \{\ \textbf{return } (\max_{r \in \mathcal{R}_A} P(r|L) > \lambda_r) \textbf{ or } (|L| > \lambda_{MQ});\ \}$$

According to this policy, we stop when the probability of the best rule overcomes a threshold λ_r or just run out of a "budget" of λ_{MQ} membership queries allocated for learning the rule with this worker.

4.1 A probabilistic quality model for wrappers

We now present our Bayesian model for estimating the probability $P(r|v_A^l, L)$ of each candidate rule $r \in \mathcal{R}_A$ of being a correct extraction rule of A for the whole set of input pages U, given a new labeled value v_A^l and the t.s. L. Our model considers noisy workers making independent and random mistakes (that is, providing erroneous labels) with error rate η.

The probability of correctness of an extraction rule is computed whenever the worker provides a new labelled value v_A^l, which will expand the current t.s. L. The posterior probability $P(r|v_A^l, L)$ can be obtained starting from the probability $P(r|L)$ by means of a Bayesian update.

The whole process is triggered by a prior p.d.f. $\mathcal{P}(r)$ over the candidate extraction rules $r \in \mathcal{R}_A$ extracting the initial annotated value of A from the input pages U. We use a simple uniform prior: $\mathcal{P}(r) = \frac{1}{|\mathcal{R}_A(U)|}$.

By applying Bayes' theorem:

$$P(r|v_A^l, L) = \frac{P(v_A^l|r, L)P(r|L)}{P(v_A^l|L)} \tag{1}$$

where $P(v_A^l|r, L)$ is the likelihood of acquiring the labeled value v_A^l conditioned to the correctness of r, once a t.s. L has been observed, and $P(v_A^l|L)$ is a *normalization factor* that can be expressed as:

$$\sum_{r_i \in \mathcal{R}_A} P(v_A^l|r_i, L)P(r_i|L),$$

to sum up all the probabilities to one.

The p.d.f. $P(v_A^l|r, L)$ can be obtained by introducing a probabilistic *generative model* to abstract the actual process leading to the generation of every possible t.s. in presence of a correct rule r. Notice that the labeled values forming the t.s. L will be labeled as either positive or negative based on the values of A, assumed correctly extracted by r, but these values are not known in advance.

We adopt a simple generative model of the labeling process that randomly chooses, without replacement, the next queried value among the set of candidate values, i.e., the next value is chosen from the set $V_A \setminus L$.

To take into account the errors of workers, we assume that a worker makes independent random mistakes, as for example in the *Classification Noise Process* [1], with an expected error rate η. It follows:

$$P(v_A^l|r, L) = \begin{cases} \frac{1-\eta}{|V_A \setminus L|}, & \text{iff } v_k \in V_A^l(r) \\ \frac{\eta}{|V_A \setminus L|}, & \text{iff } v_k \in V_A^{-l}(r) \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

where, given a correct rule r, $V_A^l(r)$ denotes the set values that can form new values labeled l, having observed the t.s. L; $-l$ is the opposite label of l.

It is worth observing that our probabilistic model depends neither on $\mathcal{R}_A$ nor on the formalism used to specify the extraction rules, and can be easily extended to other classes of extraction rules. It assumes that $\mathcal{R}_A$ contains the correct rule. This is a fairly reasonable approximation of the reality with large data-intensive websites. Anyway, if the correct rule did not exist in $\mathcal{R}_A$, ALF_η would approximate it with the rule, within the set of candidates $\mathcal{R}_A$, that most likely justifies the collected answers, taking into account the worker's error rate.

5 Estimating workers error rates with redundancy

Algorithm ALF_η computes the quality of an extraction rule by using answers provided by an inaccurate worker. However, its performances are strongly affected by a parameter: the expected error rate of the worker. If the worker accuracy is overestimated, i.e., the algorithm expects that the worker performs better than she really does, it poses less questions and the quality of the results can be compromised. Conversely, if the worker is underestimated, i.e., the algorithm assumes that the worker performs worse than she actually does, the cost augments without significant benefits: since the algorithm does not trust the worker, it does not weight the answers enough, and it ends up posing more questions than actually needed.

A simple technique for estimating the workers error rate is to rely on the availability of ground truth information. To check the workers performance, they are asked a number of control queries which have been already pre-labeled with the correct answers. However, this solution is expensive, because of the costs of preparing the ground truth, and because of the costs paid to the workers for answering the control queries rather than the real ones [27].

An alternative solution to evaluate workers performance without making use of any ground truth information is based on redundancy: the same tasks are assigned to several workers, and their error rate estimation relies only on their agreement with other workers. This approach is based on the assumption that independent workers make independent errors, which is indeed a realistic assumption with workers recruited on a crowdsourcing platform.[4] On the one hand, redundancy offers the advantage of eliminating the costs of the ground truth information; on the other hand, it originates the additional costs of employing multiple workers on the same task.

We now illustrate our approach to estimate error rates of the workers exploiting redundant tasks. We rely on our probabilistic quality model that is easily extended to consider training data produced by several workers: it suffices to concatenate the t.s. obtained by several workers into a single teaching sequence.

We now denote L^w the t.s. produced by a worker w, L^w_A the t.s. produced by the worker w for the attribute A.[5] Also, we generalize the notation L_A to indicate the t.s. obtained by concatenating the t.s. of several workers for the same attribute A, that is, $L_A = \uplus_{w \in \mathcal{W}_A} L^w_A$, where $\uplus$ denotes the concatenation over several sequences, and $\mathcal{W}_A$ indicates the set of workers that provided labelled values for attribute A.

Given a t.s. L_A, we compute the p.d.f. $P(r|L_A)$ by using Eq. 1 (and Eq. 2). Such a p.d.f. can then be used to compute the error rate of a worker w, as the average number of incorrect answers provided in the t.s. L^w, weighted by the probability of each answer, as follows:

$$\eta_w = \frac{\sum_{v^l_A \in L^w} \begin{cases} 1 - P(v_A|L_A) & , \text{ iff } l = + \\ P(v_A|L_A) & , \text{ iff } l = - \end{cases}}{|L^w|} \tag{3}$$

[4] Notice that this approach can be seen as a special case of the previous one, as a task with ground truth information can be seen as a redundant task solved by a *perfect* worker.

[5] In Sect. 6 we consider workers producing t.s. for several attributes in a single task.

where $P(v_A|L_A)$ is the probability that v_A is a correct value to extract for the attribute A, and, as it has been done before, it is reduced to the sum of the probabilities of the rules that extract v_A; that is: $P(v_A|L_A) = \sum_{r \in \mathcal{R}_A^v} P(r|L_A)$.

6 Adaptively recruiting additional workers

Observe the mutual dependency between Eqs. 1 and 3: the former computes a p.d.f. for the correctness of the rules, given a teaching sequence and the error rates of the workers who labeled its values; the latter computes the error rates of each worker, given the p.d.f. associated with the rules.

We can exploit such mutual dependency to dynamically choose the number of workers to be recruited for inferring the extraction rule of a single attribute. For every attribute we can submit a pair of tasks, each posing queries to infer the rules for the same target attribute. With the t.s. generated from these tasks, we can trigger an iterative process that interleaves the estimation of the error rates and the computation of the p.d.f. until a convergence criteria is satisfied, i.e., until these values do not significantly change anymore.[6] At the end of this process, which allows us to jointly estimate the workers error rates and the p.d.f. over the candidate rules, if a quality criterion is not satisfied (e.g., the most likely correct rule has a low probability of correctness), other workers can be recruited on the same attribute, until they produce a t.s. that allows the selection of the correct extraction rule.

However, on a real crowdsourcing platform it is not convenient to submit too short tasks composed of just a few membership queries related to a single attribute. As reported by Ipeirotis in his study on the Amazon Mechanical Turk (AMT) marketplace [20], 90 % of the AMT tasks give a reward of 10¢, and the estimated hourly wage is approximately $5: the typical 10¢ task requires about 75 s to be fulfilled. On average, for every attribute ALF_η poses a number of queries that are answered by an ordinary worker in around 15 s. Therefore, by following those guidelines, it is fairly reasonable to submit 10¢ tasks composed of queries related to 5 attributes.

For the sake of generality, we now present our solution for the composition of tasks assuming that each task includes N attributes. According to the considerations discussed above, in our experiments with real crowdsourcing workers, we set $N = 5$.

To guarantee a reliable estimation of the error rate (and thus of the probability of correctness of the rules) a straightforward solution is that of assembling tasks containing N attributes, and submit each task (at least) twice. However, we have experimentally observed that, due to the simplicity of our membership queries, the average error rate of real workers is rather low (around 10 %), and thus for a significant percentage of attributes even one worker suffices to select the correct rule (we report details on these experiments in Sects. 7.4 and 7.6).

Therefore, instead of redundantly submitting a whole task, we prepare tasks with a limited overlapping: each task is composed of N attributes, and only a subset of K *redundant* attributes is included also in another task. With the results of the redundant

[6] In our implementation, we stop when all the error rates do not change, in absolute value, more than $\Delta_\eta = 10^{-4}$.

portion of each task, we estimate the error rate of the worker that accomplished the task, and then we use such error rate estimation to compute the p.d.f. of the extraction rules for the remaining *non-redundant* attributes of the task.

Example 5 Consider given the set of attributes $\mathcal{A} = \{A_1, A_2, \ldots, A_{10}\}$. Assuming $N = 3$ and $K = 1$, the following tasks would be created: $t_1 = \{A_1, A_2, A_3\}$, $t_2 = \{A_1, A_4, A_5\}$, $t_3 = \{A_6, A_7, A_8\}$, $t_4 = \{A_6, A_9, A_{10}\}$. Note that A_1 and A_6 are submitted twice, in different tasks. A worker produces a sequence covering N attributes, e.g., w_1 produces $L^{w_1} = L^{w_1}_{A_1} \uplus L^{w_1}_{A_2} \uplus L^{w_1}_{A_3}$.

Given $L_{A_1} = L^{w_1}_{A_1} \uplus L^{w_2}_{A_1}$, we compute the p.d.f. $P(r|L_{A_1})$ of the redundant attribute A_1 and the error rates η_{w_1} and η_{w_2} of the involved workers. Similarly, we compute $P(r|L_{A_6})$, η_{w_3} and η_{w_4}.

These error rates are then used to recompute the p.d.f. of any redundant attribute in the same tasks, such as $P(r|L_{A_2})$, $P(r|L_{A_3})$, $P(r|L_{A_4})$, $P(r|L_{A_5})$, $P(r|L_{A_7})$.

Attributes whose inferred extraction rules do not satisfy a quality criterion are used to compose new tasks, which are submitted again to the crowdsourcing platform. The new sequences returned by these tasks feed a new estimation of the error rates of the workers that elaborated the same attributes, as well as an update of the p.d.f. of all the involved attributes.

Example 6 Continuing the previous example, suppose that the probability of correctness for attributes A_2, A_3 and A_7 do not reach the target quality. Then, in order to acquire more labeled values for these attributes, their candidate values will be used to formulate the queries of a new task $t_5 = \{A_2, A_3, A_7\}$ to be submitted to the crowdsourcing platform.

The produced t.s. $L_{A_2} = L^{w_1}_{A_2} \uplus L^{w_5}_{A_2}$ is used to recompute all the error rates of the workers directly involved on the attributes of the new task t_5, e.g., η_{w_5} has to be computed and η_{w_1} has to be recomputed. Notice also that w_2, which worked on A_2, is indirectly involved since its error rate η_{w_2} depends on $P(r|L_{A_2})$ that depends on η_{w_1}. Transitively following these dependencies, it turns out that the error rates of all the workers need to be recomputed, and hence all the p.d.f. of all attributes in these five tasks.

Listing 3 illustrates the psedo-code of the ALFRED algorithm, which implements the described approach. Here the crowdsourcing platform is modeled as a queue q abstracting a completion service to which task submissions are performed by means of a non-blocking operation $q.submitAll()$; completion notifications are provided via a blocking operation $q.take()$ that removes the next completed task from the queue, or just waits if none of the submitted task has been fulfilled yet.

Starting from the input set of attributes, a set of tasks are initially submitted to the crowd (line 2) following the redundancy scheme illustrated above: each task is composed of queries related to N attributes, with K attributes per task assigned also to another task. We model the composition of the submitted tasks in attributes by means of a bipartite graph, denoted $\mathcal{G}$ (line 7), whose nodes are either attributes or tasks, and there is an edge between a task and an attribute if and only if the task includes queries on the attribute. Figure 3 (right) shows the bipartite graph for the tasks

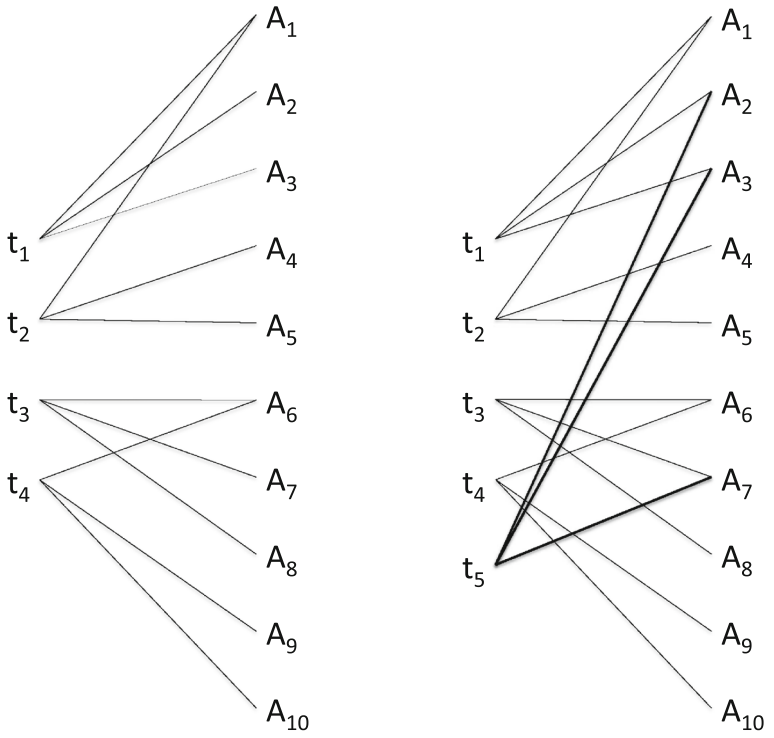

Fig. 3 The bipartite graph for the task allocation of Example 5 (*left*) and Example 6 (*right*)

described in Example 5. Observe that $\mathcal{G}$ is composed by several connected components. Each connected component includes tasks that are related by some shared redundant attribute: all the p.d.f. associated with the attributes, and all the error rates of the workers in the component, are mutually dependent. Given a task t, we denote $\mathcal{G}(t)$ the connected component of $\mathcal{G}$ that includes t (line 8).

While the queue is not empty (lines 3–20), i.e., there are tasks yet to be completed by the crowd, the first ready task t is taken from the queue, and it is added to a set of completed tasks, $\mathcal{C}$ (line 6). If all the task of the connected component $\mathcal{G}(t)$ have been already completed (line 11), the t.s. produced for the attributes of its tasks, $\mathcal{A}$, can be processed. First the workers' error rates and the p.d.f. of the associated extraction rules are computed (line 14) for the redundant attributes (i.e., attributes A such that $|\mathcal{W}_A| \geq 2$). The estimated error rates are then used to recompute the p.d.f. of the extraction rules for the remaining (non redundant) attributes of the same connected component (line 16).

All the attributes that do no reach the quality target[7] are added to a set $\mathcal{U}$ of unsolved attributes (line 17): for these attributes the collected t.s. did not lead to the production of a satisfactory extraction rule, and thus they will be added in a new task, which is

[7] We neglect any budget issue at ALFRED's level where the goal is to reach the quality target λ_r; however ALF_η bounds to λ_{MQ} the budget per attribute spent for each worker.

submitted to the crowdsourcing platform (line 19).[8] It is worth noting that as all the unsolved attributes have been already processed at least by one worker, they become redundant: they might trigger the merging of several connected components of the graph thus creating a component including a larger number of attributes.

Figure 3 (right) illustrates the graph for the tasks of Example 6: observe that task t_5, which is composed by the unsolved attributes A_2, A_3, A_7, creates a component that includes all the attributes; before t_5 submission, the same attributes were spread in two separate components.

Listing 3 ALFRED

Input: a set of attributes $\{A_1, A_2, \ldots, A_n\}$;

Parameter λ_r: target probability of correctness
Parameter λ_{MQ}: maximum budget for each worker per attribute
Parameter N: number of attributes per task
Parameter K: number of redundant attributes per task

Ensure: the set of the most probable extraction rules $r \in \mathcal{R}_A$, $\forall A \in \{A_1, A_2, \ldots, A_n\}$;

1: **let** $\mathcal{C} = \emptyset$; *// set of completed tasks*
2: $q.submitAll($ CREATETASKS$(\{A_1, \ldots, A_n\}, N, K, \lambda_r, \lambda_{MQ}))$; *// create initial tasks*
3: **while** (**not** $q.isEmpty()$) **do**
4: **let** $\mathcal{U} = \emptyset$; *// set of unsolved attributes (already with a t.s. but need more...)*
5: **let** $t = q.take()$; *// removes a fulfilled task from the completion queue*
6: $\mathcal{C} \leftarrow \mathcal{C} \cup \{t\}$; *// save as completed*
7: **let** $\mathcal{G}$ **be** *<< the bipartite assignment graph of the submitted tasks >>*;
8: **let** $\mathcal{G}(t)$ **be** *<< its connected component including t >>*;
9: **let** $\mathcal{T} \leftarrow$ *all the tasks in* $\mathcal{G}(t)$;
10: **let** $\mathcal{A} \leftarrow$ *all the attributes in* $\mathcal{G}(t)$;
11: **if** $(\mathcal{T} \subseteq \mathcal{C})$ **then**
12: *// all the tasks of the connected component that includes t have been completed*
13: **let** $R \leftarrow \{A \in \mathcal{A}$ such that $|\mathcal{W}_A| \geq 2\}$; *// redundant attributes*
14: compute $P(r|L_A)$ and η_w with Eq. 1 and Eq. 3, resp., $\forall r \in \mathcal{R}_A, \forall w \in \mathcal{W}_A, \forall A \in R$;
15: **let** $N \leftarrow \{A \in \mathcal{A}$ such that $|\mathcal{W}_A| = 1\}$; *// non-redundant attributes*
16: compute $P(r|L_A)$ with Eq. 1 with η_w as computed on line 14, $\forall r \in \mathcal{R}_A, \forall A \in N$;
17: $\mathcal{U} \leftarrow \mathcal{U} \cup \{A \in \mathcal{A}$ such that **not** GOAL$(L_A, \lambda_r, +\infty)\}$;
18: **end if**
19: $q.submitAll($ CREATETASKS$(\mathcal{U}, N, K, \lambda_r, \lambda_{MQ}))$;
20: **end while**
21: **return** $\{r \in \mathcal{R}_A$ such that $r = \text{argmax}_{r \in \mathcal{R}_A} P(r|L_A),\ A \in \{A_1, A_2, \ldots, A_n\}\}$;

7 Experimental evaluation

We have developed a working prototype that has been used to conduct experiments for evaluating the proposed approach. In this section, we first describe some details of the system prototype, then we illustrate the experimental activity that we have conducted.

[8] For the sake of simplicity we are assuming that $|\mathcal{U}|$ is a multiple of N. Otherwise, the tasks can be completed by inserting control attributes with known answers to better estimate the workers error rate.

7.1 System prototype

The prototype takes as input a collection of pages containing data of interest and one annotated page. Based on the input annotation, the system produces the initial set of candidate rules, then it generates multiple tasks, each composed by N attributes, with K redundant attributes per task, and submits them to a crowdsourcing platform.[9]

The workers recruited on the crowdsourcing platform are redirected to an interactive web application. Each worker is asked to accomplish a task, consisting of a set of membership queries actively chosen by ALF_η and posed to the worker through the web application. A page is shown to the worker, and the application asks the worker whether a proposed string occurrence, extracted by a candidate extraction rule and highlighted on the page, represents a correct value for the target attribute. To prove the task fulfillment, the worker has to insert into the crowdsourcing platform a code that the web application returns when she has completed the task.

Presenting HTML pages downloaded from an external server into a web application is not technically trivial: client side scripts and dependencies on remote resources (e.g., images) could prevent the pages to be rendered outside the server originally publishing them. To overcome these issues, our prototype integrates existing libraries that perform a server-side simulation of the execution of a browser, running also complex JavaScript and Ajax logics.[10] These libraries produce pure HTML pages, executing the JavaScript codes and replacing them with the produced outputs. To overcome possible limitations of the worker's client, when the web application of our prototype presents a query to the worker, it uses a jpeg snapshot of the original page.

It is worth observing that even if rendering a modern HTML page is CPU consuming, this task is performed on a very small number of pages, namely those that are presented to the workers (less than 10 per attribute, as illustrated by our experimental activity in Sect. 7).

7.2 Experimental evaluation

We now report the experimental evaluation that has been conducted with the prototype. First, in Sect. 7.3 we describe the dataset used for our experiments. Then, in Sect. 7.4, we present an experiment conducted with a population of real workers recruited on the crowdsourcing market. The goal of this experiment was to study how real workers behave with our particular tasks composed of a sequence of membership queries over pages from data-intensive websites. In Sect. 7.5, we illustrate experiments for evaluating ALF_η with a single noisy worker on tasks related to a single attribute. In these experiments we used synthetic workers, simulated using the results of the previous experiment, and following the *CNP* probabilistic model [1], i.e., assuming that the workers make random and independent errors with a fixed error rate. We show the impact of workers error rate on ALF_η, the algorithm driving the interaction

[9] We rely on CrowdFlower, a popular meta-platform that offers services to recruit workers on AMT.

[10] We use Selenium (http://docs.seleniumhq.org/projects/webdriver) and Phantomjs (http://phantomjs.org).

Table 1 Website and domain of each collection of pages in the first dataset; $|U|$: number of pages in the collection; $\overline{|I|}$: average number of representative pages over the attributes in the collection; $\overline{|\mathcal{R}_A|}$: average number of generated rules over the attributes in the collection

Website	Domain	$\lvert U\rvert$	$\overline{\lvert I\rvert}$	$\overline{\lvert\mathcal{R}_A\rvert}$
imdb.com	Movie	10,000	26.2	99.2
imdb.com	Actor	10,000	21.4	298.2
allmusic.com	Album	10,000	21.29	81.71
allmusic.com	Band	10,000	17.5	72.25
nasdaq.com	Stock	6,461	16.89	84.44
allgames.com	Game	10,000	39	118.33
allmovies.com	Movie	10,000	19.5	62.5
espnfc.com	Player	10,000	12	27
espnfc.com	Team	10,000	7.5	34
espon.go.com	Player	10,000	24.5	56.67
goodreads.com	Book	10,000	33.2	69.4
goodreads.com	Author	10,000	24	39

with the worker: the results motivate the introduction of our technique to estimate the workers error rate by submitting redundant tasks. Then we move to ALFRED, our system making use of redundant tasks: in Sect. 7.6 we focus on the impact of redundancy by considering tasks related to a single attribute; in Sect. 7.7, we report the results with tasks covering $N > 1$ attributes each, and then study how our algorithm performs in presence of redundancy confined to only a subset of K (with $0 \leq K \leq N$) attributes. Finally, in Sect. 7.8 we report experiments on ALFRED with real workers.

7.3 Datasets

We run our system on several collections of web pages taken from two datasets. The first one consists of 12 collections of pages from 8 websites as detailed in Table 1. For every website we randomly selected and downloaded a test set of 10,000 pages (except for nasdaq, which offers only 6,461 pages about stock quotes) and about 5 attributes, for a total of 67 attributes.

The second dataset is a subset of the public SWDE dataset [19],[11] which includes about 124,000 pages from 80 websites related to 8 different domains (10 sites per domain, 200–2,000 pages per website). We selected 34 websites (totaling 127 attributes) whose pages were still correctly rendered at the time of the submission to the crowdsourcing platform (pages from the discarded websites were not correctly rendered because their images or CSS files were no longer available).

The learning algorithms used in the evaluation were run on a representative sample set of pages selected by the PAGESAMPLER algorithm as summarized in Table 1 (column $\overline{|I|}$) for the first dataset. For the sake of space, we do not detail the same pieces of

[11] http://swde.codeplex.com.

information for the second dataset, but by averaging over all its attributes, we counted 18.39 representative sample pages and 93.07 candidate rules.

Finally, for each attribute of both datasets, we manually crafted a golden extraction rule, i.e., an XPath expression that extracts the correct values for each target attribute.[12] We compared the values extracted by the golden rules against those extracted by the rules generated by our algorithm. For every generated rule r, let $r(U)$ denote the set of values extracted from the set of pages U; we computed precision (P), recall (R), and F-measure (F) at the value level w.r.t. the corresponding golden rule r_g, as follows:

$$P = \frac{|r_g(U) \cap r(U)|}{|r(U)|}; \; R = \frac{|r_g(U) \cap r(U)|}{|r_g(U)|}; \; F = 2\frac{P \cdot R}{P + R}.$$

In the following, we measure the cost in terms of the number of membership queries, and measure the output quality in terms of the F-measure value of the produced extraction rule.

7.4 Modeling real workers

We report on a set of experiments that we conducted on a population of real workers engaged from CrowdFlower. The goal was to collect statistics about how real workers behave on our kind of tasks. Namely, we measured their error rates as they work through our web application, gaining the insights needed for setting up realistic configurations of the synthetic workers used in other experiments.

We posted on CrowdFlower 485 tasks to generate with ALF_η the extraction rules for 125 attributes, randomly selected from our datasets. Each task was paid 10¢, and posed queries to infer the rules for 5 attributes. We collected all the answers and we checked that the tasks were assigned to distinct workers. We evaluated the correctness of each answer by means of the golden rules of our datasets, and then computed the error rate of each worker as the ratio between the number of erroneous answers and the number of answers.

The observed average error rate of a real worker was $\overline{\eta} = 10\,\%$, with a standard deviation $\sigma_\eta = 14\,\%$. Interestingly, about 27 % of the workers responded correctly to all the queries.

The information gathered in this experiment has been used to set up the other experiments: the average error rate empirically observed $\overline{\eta}$ is used to set the parameter $\eta = \overline{\eta}$ in ALF_η, and the error rate distribution observed on real workers is used to create population of synthetic workers with the same error rate distribution, eventually scaled in the experiments that require populations of synthetic workers with a greater average error rate.

7.5 ALF_η evaluation

We evaluated ALF_η by conducting experiments to analyze its sensitivity to the accuracy of its parameter η as an estimation of the actual worker error rate, and to examine

[12] The datasets are available upon request.

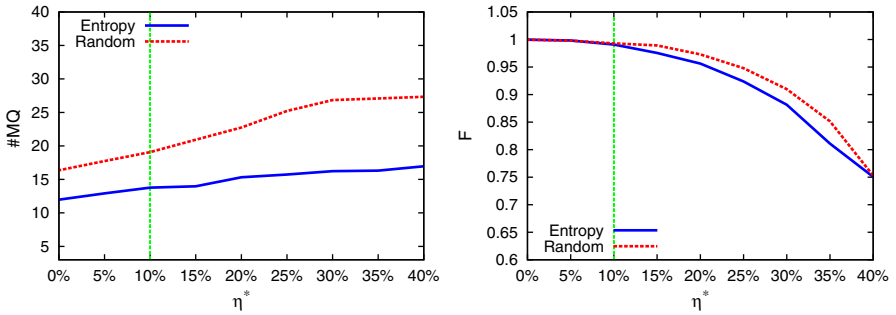

Fig. 4 ALF_η (configured with $\eta = \overline{\eta} = 10\,\%$) sensitivity to worker error rate η^*: Cost (*left*) and quality (*right*) of the output wrapper based on entropy and random query selection policy

the effectiveness of the active learning strategy adopted by the algorithm. We set up two experiments on attributes taken from our datasets by setting $\lambda_r = 90\,\%$, and $\lambda_{MQ} = +\infty$, respectively: ALF_η tries to reach the target probability of the best rule without any bound on the number of queries.

The first experiment aims at studying the effects produced by a generic inaccurate worker when ALF_η has been configured to expect a worker with the average error rate we registered on actual workers, i.e., we set $\eta = \overline{\eta} = 10\,\%$. We run ALF_η with synthetic workers whose error rate, denoted η^*, increases from 0 to 40 %. To evaluate the contributions of the active learning algorithm, we have run the same experiment using a passive learner (Random) that chooses the queries randomly.

Figure 4 reports the results of this experiment (averaged over 25 executions), and shows that as the actual error rate η^* of the workers increases, the results degrade: ALF_η poses a larger number of queries, but the quality of the output rule, in term of its F-measure, inexorably decreases. Also, Fig. 4 (left) shows the positive effects of the active learning approach: the vote entropy strategy allows the system to save several queries, especially for high error rates, without any remarkable quality loss.

The second experiment aims at empirically evaluating how an incorrect setting of the parameter η, i.e., the expected worker error rate, influences ALF_η performances. We used a synthetic worker with $\eta^* = \overline{\eta} = 10\,\%$, and repeated several inference processes, configuring ALF_η with η ranging again from 0 to 40 %.

Figure 5 reports the results (averaged over 25 executions): when the system overestimates the accuracy of worker ($\eta < \eta^*$) we observe a reduction of the number of MQ, but the quality of the output wrapper drops. The system trusts the workers and terminates quickly, thus posing less queries than actually needed. When the system underestimates the worker accuracy ($\eta > \eta^*$), some queries are wasted since the system does not trust the worker, however there is no loss in the quality of the result. For example, by setting $\eta = 40\,\%$, i.e., to a much greater value than the actual worker's error rate $\eta^* = 10\,\%$, ALF_η requires almost twice the queries that it requires when it is configured with a correct estimation of the the worker's error rate $\eta = \eta^* = 10\,\%$, but most of these queries are wasted since the F-measure gain is less than 3 %.

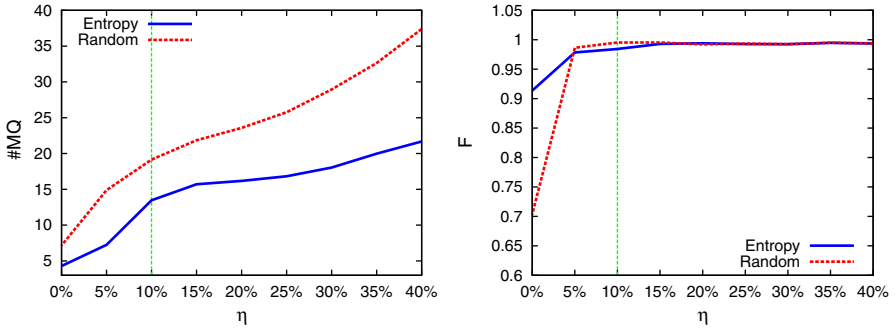

Fig. 5 ALF$_\eta$ sensitivity to η, the expected worker error rate parameter, with a noisy worker $\eta^* = \overline{\eta} = 10\,\%$: Cost (*left*) and quality (*right*) of the output wrapper based on entropy and random query selection policy

Table 2 ALFRED versus ALF$_\eta$ with a population of synthetic noisy workers; average and max total number of workers engaged per attribute ($\#w$); average F-measure of the output wrapper; average and max total number of queries ($\#MQ$); average difference between actual and estimated worker error rate ($|\eta_w - \eta^*|$); standard deviation of the output wrapper F-measure (σ_F)

	Average				Max		
	$\#w$	F	$\#MQ$	$\lvert\eta_w - \eta^*\rvert$ (%)	$\#w$	$\#MQ$	σ_F (%)
ALF$_\eta$	1	0.96	9.15	–	1	11	15
ALFRED$_{no}$	2.37	1	23.35	–	8	80	0.29
ALFRED	2.13	1	20.7	0.8	4	40	0.18
ALFRED*	2.11	1	20.27	0	4	40	0.18

Also in this experiment, the role of the active learning strategy is remarkable: w.r.t. the passive learner, the system always saves several queries. Moreover, it improves the quality of the results for small values of η, as those registered with actual workers.

7.6 Impact of redundancy

As discussed in Sect. 5, ALFRED submits the inference of the same attribute to several workers. It lazily recruits additional workers, at runtime, to estimate their error rate while minimizing the costs.

Although in practice it is convenient to assemble tasks composed of queries related to several attributes (as we discussed in Sect. 6), in this experiment we focus the study on the solely role of redundancy: we run ALFRED with tasks composed by a single redundant attribute, i.e., $K = N = 1$.

Table 2 reports the results of the experiment averaged over 20 executions in which ALFRED recruits workers from a population of synthetic workers with the same error rates distribution observed over real workers. We compare the algorithm against a baseline (ALFRED$_{no}$) in which the error rate estimation is disabled (we just set $\eta_w = \overline{\eta}$ without using Eq. 3), and against a bound (ALFRED*) in which an oracle sets $\eta_w = \eta^*$

Table 3 ALFRED: percentage of attributes (*%attr.*) that reach the target quality with 2, 3, and 4 workers; their average cost as total number of membership queries (#*MQ*)

	#workers		
	2	3	4
%attr.	89	10	1
#*MQ*	19.62	30	40

(since the workers are synthetic, their actual error rates are known). Also, in order to emphasize the impact of the redundancy, we report the performance of ALF_η, which does not rely on redundant tasks. Observe that the workers error rate estimation is precise ($|\eta_w - \eta^*| = 0.8\,\%$ when the learning terminates), and it allows the system to save queries (20.7 vs 23.35 on average). The average number of queries posed by ALFRED to learn the correct rule is very close to the lower bound set by ALFRED*. Compared to ALF_η, which employs a single worker, the number of queries is more than doubled (20.7 vs 9.15). However, notice that ALFRED always concluded the tasks with an almost perfect result, and therefore it is much more robust to variations in the workers error rates as shown by the standard deviation of the F-measure (ALFRED's $\sigma_F = 0.18\,\%$ vs ALF_η's $\sigma_F = 15\,\%$).

Overall, ALFRED was able to recruit more workers, thus paying their answers, only when it is necessary to achieve the target quality of the output wrapper. However, consider Table 3 that aggregates the same results in terms of the number of recruited workers (#workers) and the number of *MQ* posed (#*MQ*) per attribute: in a large majority of cases (89 %), ALFRED terminates recruiting only 2 workers, and seldom 3 and 4 workers are necessary (10 and 1 % of total attributes, respectively) with an average of 2.13 workers engaged and 20.7 queries posed per attribute.

So it is reasonable to conjecture that for a significant portion of the attributes requiring only 2 workers, similar results can be achieved even with less redundancy and lower costs.

The experiments presented in the next section investigate and confirm such a conjecture, thus motivating the study of a more complex scheme of redundancy than that discussed here.

7.7 ALFRED evaluation

We now present our experimental evaluation of ALFRED in the most general setting: we consider tasks composed of N attributes, each containing K redundant attributes. The goal of the experiment is to study how ALFRED is affected by the amount of redundancy initially introduced in the tasks, expressed as the value of K. Therefore we run several experiments with K ranging from 0 to N.

We analyze the cost per attribute versus the ability of ALFRED to reach the target quality independently from the noise introduced by the workers, but rather than focusing on the F-measure value of the output wrapper (which is always very close

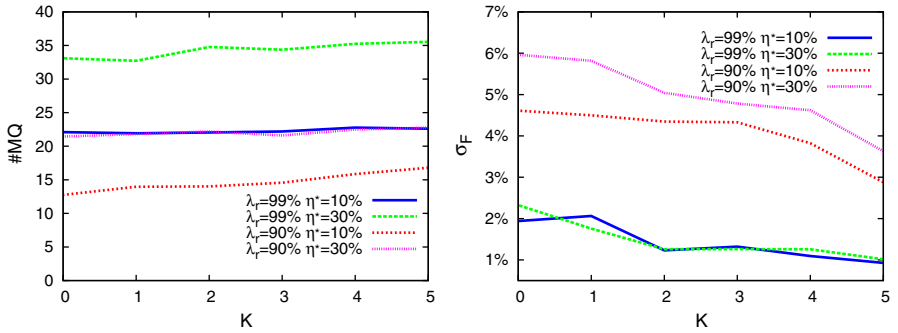

Fig. 6 The effects of the initial redundancy K: (*left*) average cost: $\#MQ$; (*right*) standard deviation of output F-measure: σ_F

to 1 in our settings), we report its average standard deviation (σ_F) over several executions, which can be considered as a measure of the predictability of the output quality of the learning process. We run the system with different populations of synthetic workers, and for each experiment we averaged the results over 100 executions. In the next section we present experiments with a population of real workers recruited on CrowdFlower, tuning the initial number of redundant attributes K according to the experiments with synthetic workers.

Figure 6 reports the results obtained with synthetic workers from two distinct populations: the first one is composed of workers with an average error rate $\eta^* = 10\,\%$, while the second one is more noisy, with $\eta^* = 30\,\%$. We also consider two different quality targets by considering $\lambda_r = 90\,\%$ and $\lambda_r = 99\,\%$.

Overall ALFRED achieves high quality ($F > 99\,\%$) with a low standard deviation ($\sigma_F < 6\,\%$) in all the configurations considered. As shown in Fig. 6 (left), ALFRED recognizes and compensates a large amount of errors (when $\eta^* = 30\,\%$) by augmenting the number of queries posed with respect to the other population (with $\eta^* = 10\,\%$).

As regards the behavior of ALFRED versus the initial amount of redundancy as K grows, Fig. 6 (left) shows that for the population of less noisy workers ($\eta^* = 10\,\%$), the cost increases from an average of around 13 ($K = 0$) to 17 ($K = 5$) queries per attribute, while σ_F, in Fig. 6 (right), decreases from 6 to 3.5 %. Both trends are less visible when $\eta^* = 30\,\%$ than when $\eta^* = 10\,\%$: With a population of noisy workers, ALFRED quickly detects that a larger amount of redundancy is needed to achieve the quality targets, and therefore the initial amount of redundancy is augmented towards the level reached at the end of the simulation, that depends on the initial values of K only loosely.

As a particular example of this behavior, consider Fig. 6 (left) when $K = 0$ and $\eta^* = 30\,\%$: this corresponds to an "optimistic" approach, in which the initial tasks are not redundant at all, and the workers error rates estimations are set with the initial fixed parameter $\eta = 10\,\%$. Even if ALF_η is overestimating the workers (the actual average error rate of the workers in this population is $\eta^* = 30\,\%$), thus giving them more trust than they deserve, the redundancy is introduced as soon as ALFRED detects that the quality targets are not reached, and it ends up with almost the same amount

Table 4 ALFRED (with $N = 5$, $K = 0$, $\lambda_r = 90\,\%$, $\eta = 10\,\%$): percentage of attributes (*%attr.*) that reach the target quality with 1, 2, 3, and 4 workers; their average cost as total number of membership queries (*#MQ*)

	#workers			
	1	2	3	4
%attr.	58	31	8	3
#MQ	7.2	16.5	26.68	37.21

of redundancy of a "pessimistic" approach in which all the attributes of the tasks are already redundant at the beginning of the simulation ($K = 5$).

It is worth observing that the quality target, i.e., the threshold λ_r, has an impact on ALFRED's effectiveness. As the plots in Fig. 6 show, increasing $\lambda_r = 90\,\%$ to $\lambda_r = 99\,\%$ pushes ALFRED to quickly increase the initial redundancy to reach the stricter quality target, and the initial redundancy K does not have a strong impact on the results. In settings with really high value of λ_r, ALFRED's sub-task redundancy does not lead to any saving in the costs.

Conversely, Table 4 has been obtained with $K = 0$, $N = 5$, $\lambda_r = 90\,\%$, and $\eta = 10\,\%$, and reports the percentage of attributes grouped by number of distinct workers employed on them. The majority of attributes (about 58 %) were assigned to only 1 worker (i.e., without redundancy), 31 % of the attributes required 2 workers, and just 11 % of the attributes needed to be assigned to more than 2 workers. The number of workers needed for an attribute was 1.57 on average, with 12.7 queries, with a significant saving compared to the simplistic redundancy scheme that allocates at least two workers for each attribute.

7.8 ALFRED on the crowd

We evaluated ALFRED with real workers recruited on the CrowdFlower crowdsourcing platform. We chose a configuration for which ALFRED produces good results in our simulations with synthetic workers, while producing a significant saving in the costs: we set $N = 5$, $K = 2$ and $\lambda_r = 90\,\%$.

These experiments have been conducted by randomly selecting 100 attributes from 35 websites within our two datasets. We repeated the experiment 4 times, in four different days, posting 135 tasks, in total. On average, to generate the extraction rules for the 100 attributes, around 34 tasks were submitted and executed by the same number of (distinct) workers. After the first submission of 25 tasks, on average only other 9 tasks have been created. Only once, a single attribute required 5 workers.

The total cost for inferring the extraction rules of each round of 100 attributes was on average $3.4, with an average cost per attribute of 3.4¢. The tasks for each round of 100 attributes were completed in 6 h, with an average quality of the output wrapper $F = 99.7\,\%$, and standard deviation $\sigma_F = 1.8\,\%$. For the whole set of experiments, workers answered to 5, 151 queries with an average number of around 37 queries per task. The average error rate observed during this session was $\overline{\eta} = 10.3\,\%$.

Table 5 Evaluation of our tasks by the CrowdFlower workers

Overall	Instructions clear	Test questions fair	Ease of job	Pay
3.8/5	4.1/5	3.8/5	3.7/5	3.6/5

CrowdFlower also provides feedbacks from the workers about the requester and the submitted jobs. Table 5 reports the scores that we obtained from the workers that fulfilled our tasks. Overall, we can conclude that the tasks were considered easy and fair by the workers, while their answers to simple queries lead to the generation of accurate wrappers.

8 Related work

Wrapper induction for extracting data from web pages has been subject of many researches for more than fifteen years [6,13]. Among the first proposals, automatic wrapper generators (e.g., RoadRunner [9], ExAlg [3]) are based on unsupervised learning techniques that do not rely on any human intervention. They can scale with the number of websites, but are generally considered too brittle to guarantee the quality needed in a production level environment. The first supervised techniques for wrapper generation (e.g., [24] and Lixto [16]) rely on a user providing feedback in the form of labeled values guiding the inference towards accurate wrappers. They can not scale and since they are intolerant towards noise in the labeling, they are not suitable for non expert users such as those engaged by a crowdsourcing platform.

More recently, a few wrapper inference approaches aiming at scaling at the Web by improving the automation level have been proposed: the system discussed in [10] automatically annotates the pages needed by a supervised technique tolerant to noise in the training data. However, it applies only for domains where it is possible to automatically obtain a set of annotations, for instance by means of lexical patterns. DIADEM [14] focuses on collections of websites in a vertical domain (e.g., real-estate), automatizing all the steps to reach the target pages containing the information and then extracting the structured data of interest. However, it strongly depends on an ontological description of the domain that has to be manually and carefully crafted by a domain expert. The latter usually cannot craft the description up-front, without iterating over a series of failing attempts that include manual problems detection and correction, over all the processing pipeline.

Similarly, [5,17–19,36] aim at scaling the extraction process on a collection of websites in a vertical domain, but require a much less sophisticated input, e.g., labeled examples to bootstrap the extraction process, or not input at all in the case of WEIR [5], which however specializes only to websites offering *redundant* data. All these proposals trade off the controllability of the extraction process for the automation level: they cannot assure the quality of the output for each attribute of every input website that the feedback, even noisy, of non-expert crowdsourcing workers can help to achieve.

The data extraction technique presented in this paper has been inspired by the seminal work of Dana Angluin [1,2], who addressed the problem of *exactly* inferring

a *concept*, i.e., a set of elements, by means of *membership queries*, i.e., question of the type "*is this an element of the target concept?*". Angluin and Laird [1] also studied the problem of learning from noisy examples, following the *CNP* probabilistic model [1], i.e., workers making random and independent errors with a fixed error rate η.

Many researchers have proposed several variations of the learning paradigm to make it practically feasible in different settings: the learning approaches in which the inference algorithm chooses the next sample to label are usually defined *active*. Active learning techniques [34] have recently gained interest as they can produce exponential improvements over the number of samples with respect to traditional supervised approaches [4].

Wrapper induction techniques that rely on active learning approaches have been already proposed in [21,32]. These studies propose a user interaction that is much more complicated than ours, since the user has to choose the correct wrapper within a set of ranked solutions. Also, they do not consider the presence of noise in the training data.

Our technique for simultaneously estimating the workers error rate, and the probability of correctness of an extraction rule is an application of the Expectation Maximization technique (EM) which has been first formalized in [11] and later widely adopted and developed [31].

The advent of crowdsourcing platforms raised new challenges. Many works have studied the problem of learning with noisy observations generated by non expert users coming from a crowdsourcing platform, e.g., [12,29,35].

Ipeirotis and co-workers [35] show that when labeling is not perfect, selective acquisition of multiple good labels is crucial and that repeated-labeling can improve label and model quality. The setting used in our experiments with real workers has been obtained by tuning a first configuration based on the parameters reported in their paper.

There have been several attempts to frame the learning from noisy workers [15,22] within theoretical guaranteed properties. These models mostly focus on the average behavior of workers and rely on strong and specific assumptions that do not hold in our setting. For instance, they allocate the same amount of redundancy for every task, and the tasks are assumed to be of the same difficulty. For other aspects these models are too generic: they rely on really mild assumptions on the workers error rate distribution, while our experiments with real workers on our specific type of tasks show that the workers error rate distribution is clearly characterizable.

Our work also focuses on the adaptive allocation of different amount of redundancy for each attribute to extract, depending on its specific difficulty and on the error rates of the recruited workers. In general, the use of redundancy raises the question of how much redundancy is actually needed to reach an optimal trade-off between costs and quality. In many situations it has been proved as good as those based on ground truth (e.g., [25]) for estimating the workers performance, with lower costs. Some theoretical bounds have been developed for specific (and simple) models (e.g., [23]) in which redundancy is established statically, i.e, before the tasks are assigned, and in a non-adaptive way, i.e., independently from their difficulty and from the provided answers. For example, Marcus et al. [30] engage 5 workers per task. Extending the

theoretical guarantees in an adaptive setting like ours appears tricky, as discussed, for instance, in [26].

A different setting from ours is also discussed in [28], where it is proposed an approach for estimating continuous quantities, e.g., the price of an item. Nevertheless they also focuses on one of the problem that we covered: how many control queries are needed to correctly evaluate workers' performance, while minimizing the costs of getting a correct estimation of the target items? More control items provide a better evaluation of the workers, but leave fewer resources for the target items that are of direct interest, and vice versa. Using their terminology, our technique to estimate workers' error rate based on workers agreement provides a *joint* estimator of the worker reliability; our technique to use the estimations obtained with the former estimator on other, non-redundant attributes, is reminiscent of their *two-stage* estimator.

9 Conclusions

We presented wrapper inference algorithms specifically tailored to exploit crowdsourcing solutions. Our approach allows the generation of wrappers by means of training data obtained by posing simple queries to workers recruited on a crowdsourcing platform. We proposed two algorithms that consider the possibility of noisy answers: ALF_η can infer a wrapper with the labeled data produced by a single worker, ALFRED can dynamically recruit multiple workers to improve the quality of the solution. We showed that ALFRED can produce high quality wrappers at reasonable costs, and that the quality of the output wrapper is highly predictable.

Our approach focuses on single value attributes. However, many sites offer pages with multivalued attributes. How to extend our algorithms to handle these pages is left to future work.

References

1. Angluin, D., Laird, P.: Learning from noisy examples. Mach. Learn. **2**(4), 343–370 (1988)
2. Angluin, D.: Queries revisited. Theor. Comput. Sci. **313**(2), 175–194 (2004)
3. Arasu, A., Garcia-Molina, H.: Extracting structured data from web pages. In: SIGMOD 2003, pp. 337–348 (2003)
4. Balcan, M.F., Hanneke, S., Vaughan, J.W.: The true sample complexity of active learning. Mach. Learn. **80**(2–3), 111–139 (2010)
5. Bronzi, M., Crescenzi, V., Merialdo, P., Papotti, P.: Extraction and integration of partially overlapping web sources. PVLDB **6**(10), 805–816 (2013)
6. Chang, C.H., Kayed, M., Girgis, M.R., Shaalan, K.F.: A survey of web information extraction systems. IEEE Trans. Knowl. Data Eng. **18**(10), 1411–1428 (2006)
7. Crescenzi, V., Merialdo, P., Qiu, D.: A framework for learning web wrappers from the crowd. In: WWW 2013, pp. 261–272 (2013)
8. Crescenzi, V., Merialdo, P., Qiu, D.: Wrapper generation supervised by a noisy crowd. In: DBCrowd, CEUR Workshop Proceedings **1025**, 8–13 (2013)
9. Crescenzi, V., Merialdo, P.: Wrapper inference for ambiguous web pages. Appl. Artifi. Intell. **22**(1&2), 21–52 (2008)
10. Dalvi, N.N., Kumar, R., Soliman, M.A.: Automatic wrappers for large scale web extraction. PVLDB **4**(4), 219–230 (2011)
11. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. J. R. Stat. Soc. B **39**(1), 1–38 (1977)

12. Doan, A., Ramakrishnan, R., Halevy, A.Y.: Crowdsourcing systems on the world-wide web. Commun. ACM **54**(4), 86–96 (2011)
13. Ferrara, E., Meo, P.D., Fiumara, G., Baumgartner, R.: Web data extraction, applications and techniques: a survey. CoRR (2012). arXiv:1207.0246
14. Furche, T., Gottlob, G., Grasso, G., Gunes, O., Guo, X., Kravchenko, A., Orsi, G., Schallhart, C., Sellers, A.J., Wang, C.: Diadem: domain-centric, intelligent, automated data extraction methodology. In: WWW 2012 (Companion Volume), pp. 267–270 (2012)
15. Gao, C., Zhou, D.: Minimax optimal convergence rates for estimating ground truth from crowdsourced labels (2013). arXiv:1310.5764
16. Gottlob, G., Koch, C., Baumgartner, R., Herzog, M., Flesca, S.: The lixto data extraction project: back and forth between theory and practice. In: PODS, pp. 1–12. ACM (2004)
17. Gulhane, P., Madaan, A., Mehta, R.R., Ramamirtham, J., Rastogi, R., Satpal, S., Sengamedu, S.H., Tengli, A., Tiwari, C.: Web-scale information extraction with vertex. In: ICDE 2011, pp. 1209–1220. IEEE Computer Society (2011)
18. Gulhane, P., Rastogi, R., Sengamedu, S.H., Tengli, A.: Exploiting content redundancy for web information extraction. PVLDB **3**(1), 578–587 (2010)
19. Hao, Q., Cai, R., Pang, Y., Zhang, L.: From one tree to a forest: a unified solution for structured web data extraction. In: SIGIR 2011, pp. 775–784. ACM (2011)
20. Ipeirotis, P.G.: Analyzing the amazon mechanical turk marketplace. XRDS **17**(2), 16–21 (2010)
21. Irmak, U., Suel, T.: Interactive wrapper generation with minimal user effort. In: WWW 2006, pp. 553–563. ACM (2006)
22. Karger, D.R., Oh, S., Shah, D.: Budget-optimal task allocation for reliable crowdsourcing systems. CoRR (2011). arXiv:1110.3564
23. Karger, D.R., Oh, S., Shah, D.: Iterative learning for reliable crowdsourcing systems. In: NIPS 2011, 1953–1961 (2011)
24. Kushmerick, N.: Wrapper induction: efficiency and expressiveness. Artif. Intell. **118**(1–2), 15–68 (2000)
25. Lee, M.D., Steyvers, M., De Young, M., Miller, B.: Inferring expertise in knowledge and prediction ranking tasks. Topics Cogn. Sci. **4**(1), 151–163 (2012)
26. Li, H., Yu, B., Zhou, D.: Error rate bounds in crowdsourcing models. CoRR (2013). arXiv:1307.2674
27. Liu, Q., Ihler, A.T., Steyvers, M.: Scoring workers in crowdsourcing: how many control questions are enough? In: NIPS 2013, 1914–1922 (2013)
28. Liu, Q., Steyvers, M., Ihler, A.: Scoring workers in crowdsourcing: how many control questions are enough? In: NIPS 2013, pp. 1914–1922 (2013)
29. Liu, X., Lu, M., Ooi, B.C., Shen, Y., Wu, S., Zhang, M.: CDAS: a crowdsourcing data analytics system. PVLDB **5**(10), 1040–1051 (2012)
30. Marcus, A., Karger, D.R., Madden, S., Miller, R., Oh, S.: Counting with the crowd. PVLDB **6**(2), 109–120 (2012)
31. McLachlan, G., Krishnan, T.: The EM Algorithm and Extensions. Wiley series in probability and statistics., vol. 2. Wiley, Hoboken (2008)
32. Muslea, I., Minton, S., Knoblock, C.A.: Active learning with multiple views. J. Artif. Intell. Res. (JAIR) **27**, 203–233 (2006)
33. Parameswaran, A.G., Dalvi, N., Garcia-Molina, H., Rastogi, R.: Optimal schemes for robust web extraction. PVLDB **4**(11), 980–991 (2011)
34. Settles, B.: Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin-Madison (2009)
35. Sheng, V.S., Provost, F., Ipeirotis, P.G.: Get another label? improving data quality and data mining using multiple, noisy labelers. In: KDD 2008, pp. 614–622, ACM (2008)
36. Wong, T.L.: Learning to adapt cross language information extraction wrapper. Appl. Intell. **36**(4), 918–931 (2012)