Wrapper Generation Supervised by a Noisy Crowd

Valter Crescenzi, Paolo Merialdo, Disheng Qiu

Dipartimento di Ingegneria
Università degli Studi Roma Tre
Via della Vasca Navale, 79 – Rome, Italy

{crescenz, merialdo, disheng}@dia.uniroma3.it

ABSTRACT

We present solutions based on crowdsourcing platforms to support large-scale production of accurate wrappers around data-intensive websites. Our approach is based on supervised wrapper induction algorithms which demand the burden of generating the training data to the workers of a crowdsourcing platform. Workers are paid for answering simple membership queries chosen by the system. We present two algorithms: a single worker algorithm (ALF$_\eta$) and a multiple workers algorithm (ALFRED). Both the algorithms deal with the inherent uncertainty of the 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 are needed. The experiments that we conducted on real and synthetic data are encouraging: our approach is able to produce accurate wrappers at a low cost, even in presence of workers with a significant error rate.

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, i.e., rules that allow the extraction of data from web pages. Supervised approaches to infer web wrappers have limited scalability, mainly because they require a set of training data, typically provided as labeled values. Unsupervised approaches (e.g. [3, 6]) have been investigated as an attempt to "scale-up" the wrapper generation process by overcoming the need of training data. Unfortunately, they have limited applicability because of the low precision of the produced wrappers.

Crowdsourcing platforms represent an intriguing opportunity to "scale-out" supervised wrapper inference approaches. These platforms support the assignment of mini-tasks to people recruited on the Web, and thus allow the engagement of a large number of workers to produce massive amounts of training data.

However, generating wrappers with the support of crowdsourcing platforms introduces a number of challenges that were not addressed in the literature: the mini-tasks submitted to the platform should be extremely simple, since they are performed by non-expert workers; their number should be minimized to contain the costs.

We are developing a framework, ALF [7], that addresses the above issues to let the crowd effectively and efficiently supervise the wrapper generation process.[1] ALF progressively infers a wrapper by posing *membership queries* (*MQ*), which are the simplest form of queries [1], since they admit only a yes/no answer (e.g., "*Is the string* `'John Wayne'` *a value to extract?*"); ALF implements an active learning algorithm to select the queries that more quickly bring to the generation of an accurate wrapper, thus reducing the costs [14]; and, finally, here we extend ALF to adopt a probabilistic model that considers errors (wrong answers) introduced by inaccurate workers [2, 15].

We have experimentally observed that with perfect workers, i.e., workers that do not make any mistake in answering the proposed membership queries, ALF generates the correct extraction rules reducing on average the number of queries by 4× with respect to a random choice [7]. However, it is well known that the workers recruited on crowd sourcing platforms are far from being perfect. On an empirical evaluation that we conducted on a popular crowdsourcing platform, we experienced a significant number of incorrect answers (around 10% on average) even for the simple membership queries posed by our system.

This paper extends our framework to manage workers that may return incorrect answers. First, we introduce ALF$_\eta$, which extends the underlying model of ALF to deal with the noisy answers of a single worker. The presence of errors introduces the challenging issue of deciding when to stop the learning process. Intuitively, when a worker is inaccurate, the costs of acquiring her answers may become not justified by the increment of quality in the inferred wrapper that these answers produce. ALF$_\eta$ needs an estimation of the worker's error rate to reason on the optimal number of queries that should be assigned to the worker. Unfortunately, at most a rough estimation of the error rate is available when the worker is engaged.

Then, to overcome this issue, we introduce ALFRED (ALF with REDundancy), an algorithm that builds on ALF$_\eta$ to find the best solution according to the training data provided by multiple workers. It adopts the conventional technique of facing the presence of errors by engaging multiple workers

[1] A demo is available at http://alfred.dia.uniroma3.it.

for solving the same tasks. However, ALFRED decides the number of workers during the learning process, at runtime, and minimizes the costs engaging only the workers actually needed to achieve the desired quality. ALFRED exploits the weighted consensus among multiple workers to estimate their error rates so that better stopping conditions can be crafted to take into account both the cost (quantified as the number of queries) and the quality of the wrapper (as estimated by a probabilistic model).

Contributions. Overall, the paper makes several contributions: (*i*) we extend our crowd based wrapper inference framework [7] in order to manage noisy answers; (*ii*) we propose a principled approach to decide at runtime how many workers should be engaged to deal with the presence of noisy answers; (*iii*) we show how to estimate the workers' error rates during the learning; (*iv*) we set several termination strategies aiming at a fair trade-off between output quality and cost; (*v*) we report a set of preliminary experimental results with both synthetic and real answers collected from the crowd.

Paper outline. The paper is organized as follows: Section 2 formalizes our setting and presents the extension to our previous probabilistic model to deal with noisy answers; Section 3 introduces ALF_η, the active learning algorithm for a single noisy worker and Section 4 presents ALFRED, its generalization to multiple workers. Section 5 reports the experimental results. Section 6 discusses related work, and Section 7 concludes the paper.

2. MODELING WRAPPER QUALITY AND NOISY WORKERS

We focus on 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 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. A rule r over the pages in U returns the ordered set of values $r(p_1), \ldots, r(p_n)$ and represents a concrete tool to build a vector of values, denoted by $r(U)$, indexed by the pages of U.

We propose a wrapper induction process that requires as input the set U of pages to be wrapped, and only a single initial annotation v_0 (which is assumed correct) of the attribute value to extract.[2]

The inference process starts by generating a space of hypothesis, i.e., a set $\mathcal{R}_{v_0}$ of candidate rules that extract the given initial annotations v_0. We consider extraction rules defined by means of expressions belonging to a simple fragment of XPath; namely, we use *absolute* and *relative* XPath expressions that specify paths to the leaf node containing the value to be extracted. Absolute rules specify paths that start either from the document root or from a node having an 'id'; relative rules start from a *template* node working as pivot. Textual leaves that occur once in every input page are considered template nodes [3].

The inference process evaluates the candidate rules in $\mathcal{R}_{v_0}$ by posing questions to workers recruited from a crowdsourcing platform: they are shown a page and asked whether a given value v from that page corresponds to a value of the target attribute. The goal is to select the rule working not only on the annotated sample page from which it has been generated, but also for all the other pages in the input collection U. Each query is formed by picking up the value in the set $V^{\mathcal{R}}_{v_0}(U)$ of values extracted from pages in U by the candidate rules $\mathcal{R}_{v_0}$.

Figure 1 shows an example: suppose that we are interested to generate a wrapper that extracts the Title from the fictional set of movie pages $U = \{p_1, p_2, p_3\}$ whose DOM trees are sketched in Figure 1(left). Assume that the initial annotation $v_0 =$'City of God' is supplied on the sample page p_1. Figure 1(right) shows the set $\mathcal{R}_{v_0} = \{r_1, r_2, r_3\}$ of candidate rules generated from this initial annotation. The queries composed by the inference process use the values $V^{\mathcal{R}}_{v_0}(U)$ that appear as elements of the vectors extracted by the rules in $\mathcal{R}_{v_0}$ from the pages in U.

The binary answer l, with $l \in \{-, +\}$, supplied by a worker adorns the queried value v with either a positive or a negative label, producing a *labeled value* denoted by v^l. An ordered set of k labeled values composes a *training sequence* (t.s.) denoted L^{k-1}, so that $L^0 = \{v_0^+\}$ and $L^{k+1} = L^k \cup \{v_k^l\}$.

Given a t.s. L^k, we introduce a probabilistic model for estimating the probability $P(r|L^k)$ of each candidate rule $r \in \mathcal{R}_{v_0}$ of being correct for the whole set of input pages U, and the probability $P(\overline{\mathcal{R}}_{v_0}|L^k)$ that the correct rule is not present in $\mathcal{R}_{v_0}$. These probabilities are updated after each new labeled value v_k^l is observed, i.e., a worker labels with l the value provided by a *MQ* on v_k and the t.s. is expanded to $L^{k+1} = L^k \cup \{v_k^l\}$.

Bayesian update rules compute the posterior probabilities $P(r|L^{k+1})$ and $P(\overline{\mathcal{R}}_{v_0}|L^{k+1})$ starting from the probabilities, $P(r|L^k)$ and $P(\overline{\mathcal{R}}_{v_0}|L^k)$ respectively, available before observing v_k^l. The process can be repeated treating each posterior probability as the priors required by the next iteration. The whole process is triggered using as a base case the *priors* $\mathcal{P}(\mathcal{R}_{v_0})$ of having generated the correct rule in $\mathcal{R}_{v_0}$, and the probability $\mathcal{P}(r)$ that r is a correct rule. Assuming that $\mathcal{R}_{v_0}$ is generated from a class of rules sufficiently expressive to include the target rule, we can fix $\mathcal{P}(\overline{\mathcal{R}}_{v_0}) = 0$,[3] and uniformly set $\mathcal{P}(r) = \frac{1}{|\mathcal{R}_{v_0}|}$.

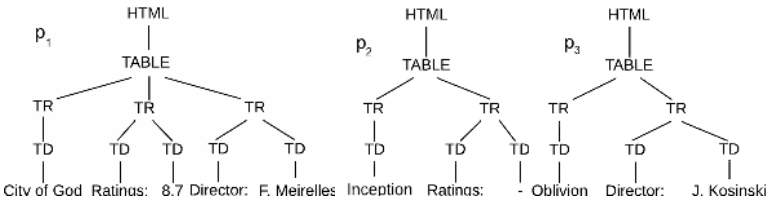
Bayesian update rules require the definition of the p.d.f. $P(v_k|r, L^k)$. This is usually obtained by introducing a probabilistic *generative model* to abstract the actual process leading to the generation of t.s.. For the sake of simplicity, we adopt the *Classification Noise Process* (*CNP*) model [2] to describe inaccurate workers that may produce independent and random mistakes with an expected error rate η, and we assume that the next value to query is randomly chosen according to a uniform p.d.f. over all values $V^{\mathcal{R}}_{v_0}(U) \setminus L^k$.

Let $P_\eta(\cdot)$ denote a p.d.f. over all possible t.s. in presence of noise, it follows:

$$P_\eta(v_k^l|r, L^k) = P(v_k^l|r, L^k) \cdot (1 - \eta) + P(v_k^{-l}|r, L^k) \cdot \eta \qquad (1)$$

[2]Such input annotation may be supplied either manually or automatically by looking up in the page a golden value from an available database. The approach can be easily generalized to deal with multiple initial annotations.

[3]In [7], where we study the possibility of dynamically tuning the expressiveness of the class, we have developed the opposite assumption.



rules \ pages		U		
		p_1	p_2	p_3
$\mathcal{R}_{v_0}$	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(.,"Ratings:")]/../../tr[1]/td
r_3 =//td[contains(.,"Director:")]/../../tr[1]/td

Figure 1: (Left) DOM of three sample pages; (Right) Extraction rules in $\mathcal{R}_{v_0} = \{r_1, r_2, r_3\}$ with v_0 ='City of God' and the set of values $V^{\mathcal{R}}_{v_0}(U)$ they extract from pages in $U = \{p_1, p_2, p_3\}$.

where $(1-\eta)$ is the probability that a worker correctly labels with l the provided value v_k, and η is the probability that she wrongly provides the opposite label, here denoted by $-l$. $P(v_k^l|r, L^k)$ is the corresponding noise-free p.d.f. and can be expressed as in [7]:

$$P(v_k^l|r, L^k) = \begin{cases} \frac{1}{|V^{\mathcal{R}}_{v_0}(U)|-|L^k|} & \text{, iff } v_k \in V^l(r) \setminus L^k \\ 0 & \text{, otherwise} \end{cases}$$

where $V^l(r)$ is the subset of values in $V^{\mathcal{R}}_{v_0}(U)$ that should be labeled l if r is the correct rule. In our generative model the values composing the t.s. L^k cannot be queried again, therefore $V^{\mathcal{R}}_{v_0}(U) \setminus L^k$ is the set of values that can be used to generate new queries.

As we discuss in the next section, these probabilities are used to effectively choose the next question, and to establish a termination condition for the learning process.

3. LEARNING EXTRACTION RULES

The probabilistic model developed in the previous section aims at computing, observed a t.s. L^k, the probability $P(r|L^k)$ that a given extraction rule r within a set of candidate rules $\mathcal{R}_{v_0}$ is correct. In this section we present ALF$_\eta$, an active learning algorithm that exploits these probabilities to minimize the number of queries to the crowd workers.

Listing 1 ALF$_\eta$: Active Learning Algorithm for a Single Noisy Worker

Input: a set of pages U
Input: the set of candidate rules $\mathcal{R}_{v_0}$
Input: a worker w and its associated error rate η_w

Output: a teaching sequence L^k

1: **let** $k = 1$; **let** $L^1 = \{v_0^+\}$;
2: **while** (**not** HALT$_{\text{ALF}}(L^k)$) **do**
3: $v_k \leftarrow$ CHOOSEQUESTION(L^k);
4: $l \leftarrow$ GETANSWER(w, v_k);
5: $L^{k+1} \leftarrow L^k \cup \{v_k^l\}$;
6: compute $P(r|L^{k+1})$, $\forall r \in \mathcal{R}_{v_0}$;
7: $k \leftarrow k + 1$;
8: **end while**
9: **return** L^k;

Listing 1 contains the pseudo-code of the ALF$_\eta$ algorithm: it processes a t.s. L^k built by actively asking to a worker (here modeled by means of the subprogram GETANSWER()) the label of a value chosen by the subprogram CHOOSEQUESTION(); ALF$_\eta$ computes a p.d.f. describing the probability of correctness over the rules in $\mathcal{R}_{v_0}$.

In every iteration (lines 2–8), the worker is asked to label a new value v_k (lines 3–4) and the t.s. is expanded (line 5). Then the probability $P(r|L^{k+1})$ is updated (line 6).

CHOOSEQUESTION() selects the next value to be labeled by the worker, i.e., the next membership query. The chosen value is that on which rules most disagree, appropriately weighted according to their probability. This is equivalent to compute the *vote entropy* [14] for each $v \in V^{\mathcal{R}}_{v_0}(U)$:

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

where: $P(v^+|L^k) = \sum_{r \in \mathcal{R}_{v_0} : r(p_v) = v} P(r|L^k)$

and $P(v^-|L^k) = \sum_{r \in \mathcal{R}_{v_0} : r(p_v) \neq v} P(r|L^k)$

are the probabilities that v is respectively either a value to extract or an incorrect value (p_v denotes the page containing v). Intuitively, the entropy measures the uncertainty of a value and querying the value with the highest entropy removes the most uncertain value:

CHOOSEQUESTION(L^k) { **return** argmax$_{v \in V^{\mathcal{R}}_{v_0}(U)} H(v)$; }

Since different termination policies can be appropriate depending on the budget constraints and on the quality targets, we propose several implementations of HALT$_{\text{ALF}}(L^k)$.

HALT$_r$: A simple policy is to stop when the probability of the best rule overcomes a threshold λ_r:

HALT$_r(L^k)$ { **return** $(\max_{r \in \mathcal{R}_{v_0}} P(r|L^k) > \lambda_r)$; }

The main limitation of this strategy is that it does not take into account the costs.

HALT$_{MQ}$: A simple policy to upper bound the costs is to stop as soon as the algorithm runs out of a "budget" of λ_{MQ} membership queries.

HALT$_{MQ}(L^k)$ { **return** $(|L^k| > \lambda_{MQ})$; }

The main limitation of this strategy is that it does not consider the quality of the output rules.

HALT$_H$: A trade-off between quality and cost can be set by posing only queries that contribute enough to the quality of the inferred rules, and stopping as soon as the costs are considered not correctly rewarded by the increment of quality in the output rules. It turns out that this can be easily modeled in term of the maximum entropy. HALT$_H$ terminates as soon as the maximum entropy of the values is below a threshold λ_H, i.e., no value is uncertain enough to deserve a query:

HALT$_H(L^k)$ { **return** $(\max_{v \in V^{\mathcal{R}}_{v_0}(U)} H(v) < \lambda_H)$; }

Whichever is the termination policy adopted by ALF_η, its efficacy in achieving an optimal trade-off between quality and the costs for the learning tasks is strongly affected by a correct estimation of the worker's error rate, η. An incorrect evaluation of η can lead to a waste of queries (when the error rate is overestimated), or to a quality loss (when it is underestimated), as confirmed by the experiments with different termination policies (reported in Section 5.2).

In our experimental evaluation, we use as a termination policy the following combination:

$$\text{HALT}_{\text{ALF}_\eta}(W, L^k) = \text{HALT}_H(L^k) \textbf{ or } \text{HALT}_{MQ}(L^k).$$

This policy leverages the same trade-off between quality and cost as in HALT_H, but focuses on the cost side by limiting through HALT_{MQ} the budget allocated for each worker.

4. INFERENCE WITH A NOISY CROWD

We now introduce another algorithm, ALFRED, that follows the conventional approach based on redundancy [15] to deal with inaccurate workers. ALFRED improves ALF_η robustness by dynamically recruiting additional workers to whom it dispatches redundant tasks. It combines the answers provided by a set W of multiple workers on the same task to estimate the workers' error rate, as well as to find the most likely rule, at the same time.

Our probabilistic model can easily deal with multiple workers by appending the t.s. L_w of every worker $w \in W$ into a unique t.s., denoted by $L = \uplus_{w \in W} L_w$, which is then used to train ALF_η. However, this approach raises two issues: (i) it is not clear how many workers should be involved in the learning process to achieve a good trade-off between output quality and cost; (ii) a correct estimation of the workers' error rates strongly affects ALF_η performances, and each worker could have an error rate radically different from others.

To overcome these issues, ALFRED dynamically chooses the amount of redundancy needed, and it exploits the cumulative knowledge both to find the solution on which most of the weighted workers consensus converges, and to estimate the workers' error rates.

Listing 2 illustrates ALFRED pseudo-code: the main loop (lines 2–15) alternates two phases. First, the workers are engaged (lines 3–8): each worker w trains an ALF_η instance, thus producing a t.s. L_w. In this phase, the worker is associated with an initial error rate (line 5).[4] The second phase (lines 10–14) starts as soon as the first phase has accumulated enough workers (at least W_0). The goal of this phase is to leverage all the t.s. provided by the workers in order to compute both the p.d.f. of the rules $P(r|L)$ and the individual error rate η_w of each worker $w \in W$.

Our solution is inspired by the work on *Truth Finding Problems* [10], and exploits the mutual dependency between the probability of correctness of the rules and the error rates of the workers answering the *MQ*. The error rate is defined in term of probability as follows:

$$\eta_w = \frac{\sum_{v_k^l \in L_w} \begin{cases} 1 - P(v_k^+|L) & \text{, iff } l = + \\ P(v_k^+|L) & \text{, iff } l = - \end{cases}}{|L_w|}$$

[4]We use the average error rate empirically estimated in our experiment with real workers; another option is to derive this value from the workers ranks provided by the crowdsourcing platform.

Listing 2 ALFRED: Active Learning Algorithm with Multiple Noisy Workers

Input: a set of pages U
Input: the set of candidate rules $\mathcal{R}_{v_0}$

Parameter: number of initial workers W_0
Parameter: a threshold λ_η for error rates convergence

Output: the most probable extraction rule $r \in \mathcal{R}_{v_0}$

```
1:  let W = ∅; // set of engaged workers
2:  repeat
3:    repeat
4:      let w = ENGAGEWORKER();
5:      let η_w = GETERRORRATE(w);
6:      let L_w = ALF_η(U, R_{v0}, w);
7:      W ← W ∪ {w};
8:    until (|W| < W_0); // wait for workers
9:    let L = ⊎_{w∈W} L_w; // append all t.s.
10:   repeat
11:     compute P(r|L) by using η_w, ∀r ∈ R_{v0} ;
12:     let η_w^prev = η_w, ∀w ∈ W ; // save previous η
13:     compute η_w by using P(r|L), ∀w ∈ W;
14:   until (Σ_{w∈W}(η_w − η_w^prev)^2 > λ_η · |W|);
15: until (HALT_ALFRED(W, L));
16: return argmax_{r∈R_{v0}} P(r|L);
```

and it can be interpreted as the probability of the worker of correctly answering a *MQ*, estimated by averaging over all the *MQ* in the t.s. L_w that she has provided. Since also the probability is defined in term of error rates, as shown in Eq. 1, their computation is interleaved (lines 11 and 13) until their values do not significantly change anymore.[5]

The termination condition of ALFRED can be set according to different policies by specifying $\text{HALT}_{\text{ALFRED}}$. In our implementation, in order to take into account both budget constraints and quality targets, we used the following combination:

$$\text{HALT}_{\text{ALFRED}}(W, L) = \text{HALT}_r(L) \textbf{ or } \text{HALT}_{MQ}(L)$$

Observe that the recruitment of new workers negatively influences the latency of the system. A strategy to contain the latency of the system is to recruit bulk workers, but this is beyond the scope of the present paper.

5. EXPERIMENTAL EVALUATION

We use a dataset composed of 5 collections of pages: actor and movie pages from www.imdb.com; band and album pages from www.allmusic.com; stock quote pages from www.nasdaq.com. For each collection we selected about 10 attributes, for a total of 40 attributes. Then we manually crafted a golden XPath rule for each attribute. We randomly selected a training sample set of 500 pages from each collection of pages, and another (disjoint) test set of 2,000 pages. Our algorithm was run on the training sample, and the output extraction rules were evaluated on the test set. We compared the (non-null) values extracted by the golden rules against those extracted by the rules generated by our

[5]We have empirically observed the convergence of the algorithm. A formal proof is left as future work.

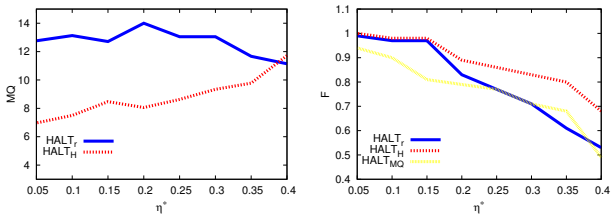


Figure 2: ALF$_\eta$ ($\eta = 0.09$) sensitivity to worker error rate η^*: Cost (left) and quality (right) of the output wrapper.

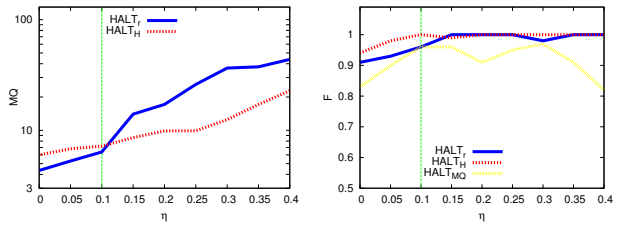


Figure 3: ALF$_\eta$ sensitivity to the expected worker error rate η with a noisy worker $\eta^* = 0.09$: Cost (left) and quality (right) of the output wrapper.

algorithm. For every generated rule r, and given a test 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: $P = \frac{|r_g(U) \cap r(U)|}{|r(U)|}$; $R = \frac{|r_g(U) \cap r(U)|}{|r_g(U)|}$; $F = 2\frac{P \cdot R}{P+R}$.

We report the results of two sets of experiments: the first one was conducted to test ALF$_\eta$ with a single worker, in the second set of experiments we consider ALFRED in presence of multiple workers. We evaluate our algorithms by using synthetic workers following the *CNP* probabilistic model [2], i.e., workers making random and independent errors with a fixed error rate η^*.

In order to estimate the error rate distribution over a population of real workers, we also conducted a preliminary experiment with workers engaged by means of CrowdFlower, a meta platform that offers services to recruit workers on AMT.

5.1 Experiments with AMT Workers

The main intent of this preliminary experiment was to evaluate the error rate distribution of a population of real workers recruited on a crowdsourcing platform. We submitted 100 tasks to 100 distinct workers. Each task was paid 10¢ and consisted on a set of 20 *MQ* to generate the extraction rules for several attributes of our dataset. The ground truth was straightforwardly obtained by the results of our golden extraction rules. We used ALF$_\eta$ configured with HALT$_r$ as termination condition ($\lambda_r = 0.8$), and with $\eta = 0.1$.

The average error rate of an AMT worker was $\widehat{\eta} = 0.09$, with a standard deviation of $\widehat{\sigma}_\eta = 0.11$. About 1/3 of the workers responded correctly to all the queries, and the response time for each *MQ* was around 7 seconds. On average the number of *MQ* posed by the system to infer a rule was 4, and each task contained enough queries to learn 5 rules.

The information obtained from this experiment was then used to set up realistic configurations of the synthetic workers for the other experiments: the average error rate empirically observed $\widehat{\eta}$ is used to set ALF$_\eta$'s parameter $\eta = \widehat{\eta}$ as well as the initial worker error rate estimation of ALFRED, $\eta_w = \widehat{\eta}$; also, the synthetic workers used to test ALFRED are created using the same error rate distribution observed on real workers.

5.2 Single Noisy Worker

The main goal of the experiment was to evaluate the effects of the workers' mistakes on ALF$_\eta$, our algorithm in absence of redundancy. In figure 2 we show the effects of an inaccurate worker with different termination strategies, by setting $\eta = \widehat{\eta} = 0.09$. We simulated workers with an error rate η^* increasing from 0.05 to 0.4. The thresholds of the three termination conditions considered, HALT$_r$, HALT$_{MQ}$ and HALT$_H$ were set to $\lambda_r = 0.8$, $\lambda_{MQ} = 5$ and $\lambda_H = 0.2$, respectively.

As the error rate η^* of the worker increases, the results degrade with every termination strategy, as expected. However, HALT$_H$ detects the wider presence of uncertain values, and tries to compensate with a greater number of queries; conversely, since HALT$_r$ focuses only on the most likely rule, it poses a rather steady number of queries, and the output quality is more seriously compromised.

We then empirically evaluated how an incorrect setting of the parameter η, i.e., the expected worker error rate, influences ALF$_\eta$ performances. We used a single worker with $\eta^* = \widehat{\eta} = 0.09$, and repeated several inference processes, configuring ALF$_\eta$ with η ranging from $\eta = 0$ to 0.4 as reported in Figure 3.

When the system overestimates the accuracy of worker ($\eta < \eta^*$) we observe a reduction of the number of *MQ*, but the quality of the wrapper drops. The system trusts the workers and terminates quickly, thus posing less questions than actually needed. When the system underestimates the worker accuracy ($\eta > \eta^*$), some *MQ* are wasted since the system does not trust the worker. With an η larger than η^* by +0.3, HALT$_r$ requires more than 40 *MQ*, i.e., 5× those required when $\eta = \eta^*$. Observe that many *MQ* are wasted since the F-measure gain is less than 5%.

	average			max		
	$\|W\|$	F	$\#MQ$	$\|W\|$	$\#MQ$	σ_F
ALF$_\eta$	1	0.92	7.58	1	11	0.17

Table 1: ALF$_\eta$ inference with synthetic workers

Table 1 reports ALF$_\eta$ results when the termination policy HALT$_{\text{ALF}_\eta}$ has been instantiated by setting the parameters $\lambda_H = 0.2$ and $\lambda_{MQ} = 10$. ALF$_\eta$ requires just a few queries ($\#MQ = 7.58$) to learn rather accurate wrappers ($F = 0.92$). However, there is a significant standard deviation ($\sigma_F = 17\%$) in the output quality that makes the algorithm not that robust to workers' errors.

5.3 Multiple Noisy Workers

As discussed in Section 4, ALFRED builds on ALF$_\eta$, and recruits additional workers to estimate their error rate and to find the correct rule at the same time. We rely on the termination strategy of the outer algorithm (HALT$_{\text{ALFRED}}$) to achieve the target quality, while for the inner ALF$_\eta$ instance we use the same termination policy (HALT$_{\text{ALF}_\eta}$) focused on

	average				max		
	$\|W\|$	F	$\#MQ$	$\|\eta_w - \eta^*\|$	$\|W\|$	$\#MQ$	σ_F
ALFRED$_{no}$	2.33	1	18.6	—	9	83	0.01
ALFRED	2.07	1	16.1	0.8%	4	44	0.01
ALFRED*	2.05	1	16.07	0%	4	40	0.01

Table 2: ALFRED inference with synthetic workers

the costs as in the previous experiment. To evaluate ALFRED performances, we organized as many tasks as the number of attributes of our experiment (40). For each task we executed ALFRED (with $\lambda_\eta = 10^{-4}$) by recruiting workers from a virtual population with the same error rate distribution observed over the real workers (the results have been averaged over 20 executions).

ALFRED's results in Table 2 demonstrate the role played by the workers' error rate estimation. We compare the algorithm against a baseline (ALFRED$_{no}$) in which the error rate estimation is disabled (we just set $\eta_w = \widehat{\eta}$), and against a bound (ALFRED*) in which an oracle sets $\eta_w = \eta^*$. The workers' error rate estimation is precise ($|\eta_w - \eta^*| = 0.8\%$ when the learning terminates), and it allows the system to save queries (16.1 vs 18.6 on average). The average number of *MQ* posed by ALFRED to learn the correct rule is only a negligible amount larger than the lower bound set by ALFRED*. The costs are more than twice those paid running ALF$_\eta$ with a single worker (16.1 vs. 7.58). However, notice that ALFRED always concluded the tasks with a perfect result, and that it is robust to workers' error rates ($\sigma_F = 1\%$).

ALFRED terminates in most of the cases (94%) engaging only 2 workers, and seldom recruited 3 and 4 workers (5% and 1%, respectively). Overall, ALFRED was able to recruit more workers, thus paying their answers, only when needed to achieve the target quality of the output wrapper.

6. RELATED WORK

Wrapper induction for extracting data from web pages has been subject of many researches [5]. A wrapper inference approach tolerant to noise in the training data has been proposed in [8], however it applies only for domains where it is possible to automatically obtain a set of annotations.

Active learning approaches [14] have recently gained interest as they can produce exponential improvements over the number of samples wrt traditional supervised approaches [4]. The advent of crowdsourcing platforms has led to new challenges. The main issue is to learn from noisy observations generated by non expert users.

Wrapper induction techniques that rely on active learning approaches have been proposed in [11, 13]. These studies rely on a more complicated user interaction 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. Many works have studied the problem of learning with noisy data coming from crowdsourcing platform workers, e.g., [9, 12, 15]. [15] shows that when labeling is not perfect, selective acquisition of multiple good labels is crucial and that repeated-labeling can improve label quality and model quality. [12] proposes a crowdsourcing assisted system, CDAS, for data analytics tasks; the system faces the presence of noisy answers by submitting redundant tasks and adopts a voting strategy to estimate the correct solution. Compared to our approach, the number of workers as well as their accuracies are statically determined, based on the workers' historical performances.

In our previous work [7], we studied wrapper inference with a single and perfect worker.

7. CONCLUSIONS

We presented wrapper inference algorithms specifically tailored for working with the support of crowdsourcing platforms. Our approach allows the wrappers to be generated by posing simple questions, membership queries, to workers engaged on a crowdsourcing platform. We proposed two algorithms that consider the possibility of noisy answers: ALF$_\eta$ recruits a single worker, ALFRED can dynamically engage 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.

8. REFERENCES

[1] D. Angluin. Queries revisited. *Theor. Comput. Sci.*, 313(2):175–194, 2004.

[2] D. Angluin and P. Laird. Learning from noisy examples. *Mach. Learn.*, 2(4):343–370, Apr. 1988.

[3] A. Arasu and H. Garcia-Molina. Extracting structured data from web pages. In *SIGMOD 2003*.

[4] M.-F. Balcan, S. Hanneke, and J. W. Vaughan. The true sample complexity of active learning. *Machine Learning*, 80(2-3):111–139, 2010.

[5] C.-H. Chang, M. Kayed, M. R. Girgis, and K. F. Shaalan. A survey of web information extraction systems. *IEEE Trans. Knowl. Data Eng.*, 18(10):1411–1428, 2006.

[6] V. Crescenzi and P. Merialdo. Wrapper inference for ambiguous web pages. JAAI, 22(1&2):21–52, 2008.

[7] V. Crescenzi, P. Merialdo, and D. Qiu. A framework for learning web wrappers from the crowd. In *WWW 2013*.

[8] N. N. Dalvi, R. Kumar, and M. A. Soliman. Automatic wrappers for large scale web extraction. *PVLDB*, 4(4):219–230, 2011.

[9] A. Doan, R. Ramakrishnan, and A. Y. Halevy. Crowdsourcing systems on the world-wide web. *Commun. ACM*, 54(4):86–96, Apr. 2011.

[10] X. Dong, L. Berti-Equille, Y. Hu, and D. Srivastava. Solomon: Seeking the truth via copying detection. *PVLDB*, 3(2):1617–1620, 2010.

[11] U. Irmak and T. Suel. Interactive wrapper generation with minimal user effort. In *WWW 2006*.

[12] X. Liu, M. Lu, B. C. Ooi, Y. Shen, S. Wu, and M. Zhang. Cdas: a crowdsourcing data analytics system. In *VLDB 2012*.

[13] I. Muslea, S. Minton, and C. A. Knoblock. B. Settles. Active learning with multiple views. *JAIR 2006*.

[14] Active learning literature survey. CS Tech. Rep. 1648, University of Wisconsin–Madison, 2009.

[15] V. S. Sheng, F. Provost, and P. G. Ipeirotis. Get another label? Improving data quality and data mining using multiple, noisy labelers. In *KDD 2008*.