A Framework for Learning Web Wrappers from the 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

The development of solutions to scale the extraction of data from Web sources is still a challenging issue. High accuracy can be achieved by supervised approaches but the costs of training data, i.e., annotations over a set of sample pages, limit their scalability. Crowd sourcing platforms are making the manual annotation process more affordable. However, the tasks demanded to these platforms should be extremely simple, to be performed by non-expert people, and their number should be minimized, to contain the costs. We introduce a framework to support a supervised wrapper inference system with training data generated by the crowd. Training data are labeled values generated by means of membership queries, the simplest form of queries, posed to the crowd. We show that the costs of producing the training data are strongly affected by the expressiveness of the wrapper formalism and by the choice of the training set. Traditional supervised wrapper inference approaches use a statically defined formalism, assuming it is able to express the wrapper. Conversely, we present an inference algorithm that dynamically chooses the expressiveness of the wrapper formalism and actively selects the training set, while minimizing the number of membership queries to the crowd. We report the results of experiments on real web sources to confirm the effectiveness and the feasibility of the approach.

Categories and Subject Descriptors

H.3.5 [**Information Storage and Retrieval**]: On-line Information Services —*Web-based services*

General Terms

Algorithms, Experimentation

Keywords

wrapper generation, crowdsourcing, active learning

1. INTRODUCTION

Although many research efforts concentrated on the development of methods and tools to generate web wrappers, large-scale data extraction is still a challenging issue.

Early proposals to infer web wrappers for data intensive websites were based on supervised approaches. Wrappers were generated starting from a set of training data, typically provided as labeled values, i.e., annotated pages. To overcome the need of human intervention in the production of training data, unsupervised approaches have been investigated. They exploit the local regularities of script-generated web pages to infer a wrapper. Unsupervised approaches adopt sophisticated algorithms to generate the wrappers, and represent an attempt to "scale-up" the wrapper generation process. Unfortunately, although they eliminate the costs of training data, they have a limited applicability because of the low precision of the produced wrappers.

The recent advent of crowd sourcing platforms (such as, for example, Amazon Mechanical Turk) can open new opportunities for supervised approaches. These platforms provide support for managing and assigning mini-tasks to people. In the wrapper production process, crowd sourcing platforms can be used to produce massive training data for supervised wrapper inference systems. As they facilitate the involvement of a large number of persons to produce the training data, we may say that they represent a solution to "scale-out" the wrapper generation process. However, to obtain an efficient and effective process, two main issues need to be addressed. First, since mini-tasks are performed by non-expert people, they should be extremely simple. Second, since the costs of producing wrappers become proportional to the number of mini-tasks, the number of training data produced by the crowd to infer a wrapper should be minimized.

We are developing a system that relies on crowd sourcing platforms to create accurate web wrappers. Our system adopts a supervised approach to infer wrappers with training data generated by means of a crowd computing platform. The mini-tasks submitted to the platform consist of *membership queries* (*MQ*), which are the simplest form of queries, since they admit only a yes/no answer (e.g., "*Observe this page: is the string* `'Dean Martin'` *a correct value to extract?*") [1]. To address the costs issue, our system is able to select the queries that more quickly bring to infer an accurate wrapper, thus minimizing the number of mini-tasks assigned to the crowd platform.

The traditional approach to build wrappers for large websites is to provide training data for the attributes of interest on a set of pages, and then to apply an inference algorithm to learn a wrapper. We observe that there are two hidden assumptions behind the approach: 1) the formalism used by the learning algorithm to specify the wrapper is sufficiently expressive, 2) the wrappers inferred from the sample set, hopefully work also on the whole set of pages.

WWW 2013, May 13–17, 2013, Rio de Janeiro, Brazil.
ACM 978-1-4503-2035-1/13/05.

As the following running example illustrates, these assumptions can largely affect the learning costs and the quality of the solution.

1.1 Running Example

Suppose we are interested in extracting data about actors from the pages of a large movie website. A wrapper corresponds to a set of extraction rules, where each rule extracts the value of an attribute (e.g. the actor name). For the sake of simplicity, we concentrate on a single rule, but the discussion can be trivially extended to the more general case where a wrapper is composed of a set of rules. To illustrate the example without digging into the technical details of the formalisms used to express the extraction rules, we represent pages as tables, where data is organized in rows and columns, as shown in Figure 1(a). In such a simplified abstraction, an extraction rule specifies the cell containing the relevant data, and it can be expressed by *absolute* coordinates (e.g., first row, second column), denoted abs(*row*,*col*), or by *relative* coordinates, that is, with respect to another cell (e.g. the first cell located right of/left of/above/under the cell containing 'Home'), denoted right-of('*x*'). For example, according to *Hopkins*'s page, candidate extraction rules for Name are abs(1,1) and above('Height'). Similarly, rules for Home are abs(6,2) and right-of('Home'). More complex relative rules can be specified with a richer (more expressive) class, which admits longer paths from a pivot; e.g., below-right-of('Born'), to extract data from the cell containing the 'Latest Film'.

Later in the paper, we shall refer to a concrete representation (the same used in the implementation of the system), where pages are modeled with their DOM trees, and rules are XPath expressions.

The Expressiveness Problem. As the following example illustrates, the number of training data to correctly infer a wrapper depends on the expressiveness of the language used to specify the rules.

Let us concentrate on the extraction of actors' Name. Suppose that the learning algorithm adopts only *absolute* extraction rules: a correct rule for Name is abs(1,1). Note that only one training data would suffice to infer this rule. Suppose now to adopt a more expressive language, which also includes *relative* rules. Using just one labeled value, say `'Antony Hopkins'` on page p_h, several rules can be generated to extract the attribute Name: abs(1,1), above('Height'), above('1.74m'). To determine the correct rule it is necessary to carefully choose at least another annotation: only with the help of the labeled value `'Laura Wolf'` on page p_w we have evidence that above('Height') does not work as an extraction rule for Name.

This simple example illustrates a well known learning theory results: the more expressive is the model, the larger is the number of labeled examples that are necessary to infer the correct rule [1, 6] (intuitively, the space of hypotheses is larger and thus more examples are needed to discard the incorrect ones).

Supervised wrapper inference approaches that ignore the costs of training data tend to work with an overly expressive class of rules to have enough expressiveness for all the attributes. However, as our example emphasizes, this implies more expensive training data.

The Sampling Problem. Suppose that Home is an attribute to extract. If the sample set is composed only of awarded actors (such as *Hopkins*), the inferred rule could not work for the broader set of all actors, including those without any award (such as *Smith* and *Wolf*). For example, the rule abs(6,2) for Home might work for the awarded actors, but it does not extract the required information for others.

The usual approach to address this issue is to require a large set of labeled values that hopefully covers all the possible types of target pages. However, if the labeled values are generated by submitting a task to a crowd sourcing platform, a bigger set implies higher costs; also, the training data should be representative for the whole collection of target pages and its choice is not trivial. For some domains a set of labeled examples can be obtained automatically. For instance, if we have a small database of popular actors, we can annotate the pages when they offer data matching with those stored in the database [8]. However, the database could be biased, e.g. it contains only famous (awarded) actors, leading to the generation of wrong extraction rules, as discussed in the example.

1.2 Overview and Contributions

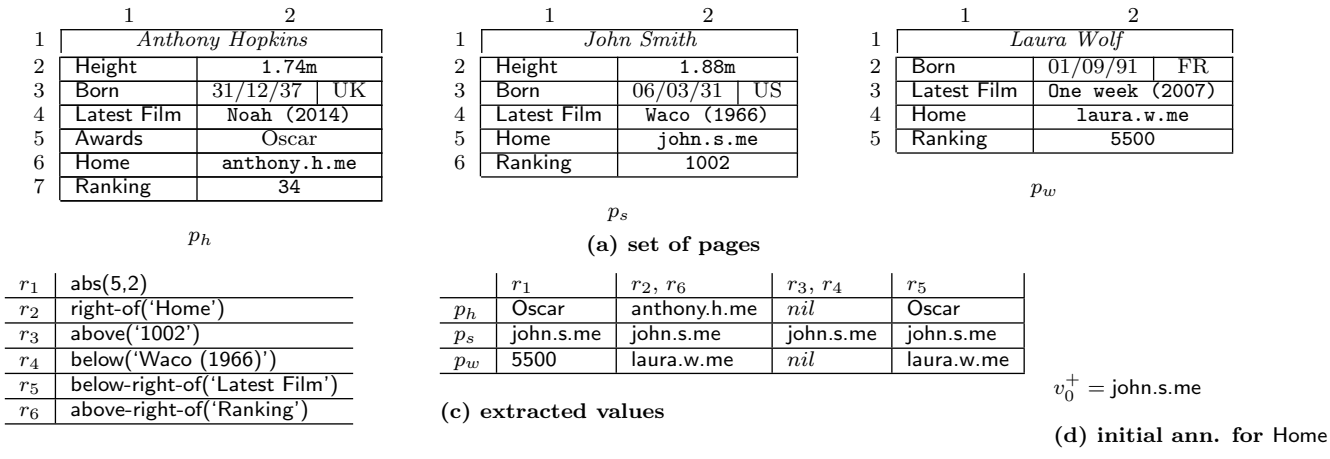
We propose a logical framework based on original solutions for exploiting crowd platforms to infer wrappers around large web sources. Since we aim at demanding to a crowd platform the burden of generating labeled examples, our approach considers a cost takes into account the number of membership queries submitted to a crowd platform.

Our framework includes a supervised active learning algorithm that aims at minimizing the number of membership queries to infer a wrapper. Since the costs of learning a wrapper depends on the expressiveness of the class of rules, unlike traditional approaches we do not work with a statically defined class, but we propose an original approach inspired to a statistical learning technique [14, 15], called *structural risk minimization* (SRM), in which the expressiveness of the language is determined at runtime. To this end, we organize the class of candidate rules into a hierarchy of classes of increasing expressiveness: initially the correct rule is searched only within the less expressive class. The class of rules is lazily expanded only if it is actually needed.

A fundamental decision is whether and when expanding the set of candidate rules: we introduce a probabilistic model that evaluates the quality of the wrapper by computing the probability that the rules in the *current* class of candidate rules are correct. Our learning algorithm exploits the probabilistic model in order to trigger the expansion only if there is enough evidence that the correct rule is not amongst the current set of candidates.

The algorithm adopts active learning techniques [13] to ask for additional labeled values: it selects a value and poses a membership query to obtain a confirmation about the correctness of the extracted value. By accurately choosing this query, the user interaction is minimized. Our experiments prove that our learning algorithm can infer high quality wrappers with a fraction of the queries required by a traditional approach.

Our algorithm infers the wrapper on a set of labeled values, and the quality model evaluates the wrapper on a larger set, ideally on the whole set of target pages. However, in many practical cases the evaluation on the whole set is unrealistic because of its size. To overcome this issue, our frame-



(a) set of pages

(b) extraction rules

(c) extracted values

(d) initial ann. for Home

Figure 1: Running Example

work also includes an algorithm to compute a small set of representative pages: the extraction rules inferred and evaluated against our representative set also work on the larger set of target pages.

Our experiments show that our algorithm is able to select a representative set several orders of magnitude smaller than the whole set of target pages, and that wrappers inferred from our representative sample outperforms (in term of precision and recall) wrappers generated from much larger randomly selected sets.

In summary, we make the following contributions:

- a framework that exploits crowd platforms to infer wrappers around large web sources;
- a cost model that takes into account both the processing costs and the human intervention costs needed to feed the crowd platform;
- a probabilistic quality model for computing correctness of a wrapper over the whole set of pages even if it is inferred on a smaller set of pages chosen by a sampling algorithm;
- an active learning algorithm for generating high quality wrappers in a cost-effective manner;
- a sampling algorithm for selecting small yet representative sets of pages;
- the results of an experimental evaluation of our framework on real life websites.

The paper is organized as follows: Section 2 formalizes our setting; Section 3 develops our probabilistic model to characterize the correctness of extraction rules; based on the model, Section 4 presents the active learning algorithm to infer extraction rules; Section 5 introduces the sampling algorithm; Section 6 discusses experiments with a set of sources from the Web; finally, Section 7 discusses related work and Section 8 concludes the paper.

2. PRELIMINARIES

Let $U = \{p_1, p_2 \dots p_n\}$ be a set of pages. Every page publishes several attributes of interest (e.g., in our running example, actors' Ranking, Height, etc.). For simplicity, we develop the discussion concentrating on one attribute, and we assume that its values are either a textual leaf of the DOM tree representation of the pages, or a distinguished *nil* value. We write $v \in p$ to denote that v is a value of the page p, and p_v to denote the page in which the value v is located.

We refer to a generic *extraction rule* (or simply *rule*) r over the set of pages U as a concrete tool to build a vector of values indexed by the pages in U such that $r(p) \in p \cup \{nil\}$. Every rule extracts one vector of values from U denoted $r(U)$. Figure 1(c) shows the vectors extracted by the rules r_1, r_2, r_3, r_4, r_5, r_6 in Figure 1(b). We denote by $R(U)$ the set of vectors obtained by applying a set of rules R over U, and blur the distinction between a rule and the vector it extracts from U. Note that $|R(U)| \leq |R|$, with the strict inequality holding whenever a vector is extracted by different rules.

We introduce the concept of *labeled sample value* (or simply *labeled value*) v^l where $v \in p_v$ is a value from a page p_v, and $l \in \{+, -\}$ is either a positive or a negative label. In the following v^+ and v^- denote a positively labeled value (or annotation) and a negative labeled value, respectively, i.e., the two possible answers to a *MQ*.

A rule r is *admissible* wrt a set of labelled values L (denoted $L(r)$) iff:

$$L(r) \;\Leftrightarrow\; \forall v^l \in L, \begin{array}{l} l = + \rightarrow r(p_v) = v \\ l = - \rightarrow r(p_v) \neq v \end{array}$$

that is, it is compliant with the labels in the set.

The concept can be trivially extended to a set of rules R. We denote by $R_L = \{r \in R : L(r)\}$ the subset of admissible rules in R wrt L, and by $\widehat{V}_L^R(U)$ all the values they extract from U: $\widehat{V}_L^R(U) = \{v : v = r(p), r \in R_L, p \in U\}$.

EXAMPLE 1. *Let p_h, p_s and p_w be the pages in Figure 1(a) and let $U = \{p_h, p_s, p_w\}$. The attribute* Home *is extracted by the rule $r_2 =$*right-of('Home')*: two positive annotations*

are v_0^+ ='john.s.me' *and* v_1^+ ='laura.w.me'*; a negative labelled value is* v_2^- ='5500'*. Observe that* r_2 *is admissible wrt* $L = \{v_0^+, v_1^+, v_2^-\}$*. Now consider another rule* r_1=abs(5,2) *and the set of rules* $R = \{r_1, r_2\}$*. Then* r_1 *is not admissible wrt* L *since* $r_1(p_w)$ ='5500' *which is the negatively labelled value* v_2^-*. Hence,* $R_L = \{r_2\}$ *and* $\widehat{V}_L^R(U) = \{v_0, v_1, v_3\}$ *where* v_3 ='anthony.h.me'.

In the following, given a set of rules R, we will only consider special ordered sets of labeled values, called *training sequences*, which are formed by an initial set of positive annotations, and then by adding only new values which are still admissible with respect to those already seen. Intuitively, a training sequence lists the answers to the *MQ* posed to learn a extraction rule.

A Training Sequence *(t.s.)* L *wrt a set of rules* R *and a set of pages* U *is specified by a sequence of labeled values that defines a sequence of (observed) sets* L^k *with* $L^{k+1} = L^k \cup \{v_k\} = \{v_0^+, \ldots, v_{a-1}^+, v_a \ldots, v_k\}$ *such that: (i) it begins with sequence of a* ≥ 1 *annotations* $v_0^+, \ldots, v_{a-1}^+ \neq nil$ *with positive labels, and (ii)* $\forall k \geq a$, $v_k \in V_{L^k}^R(U) = \widehat{V}_{L^k}^R(U) \setminus L^k$.

The constraint (i) on the first annotations of the sequence is useful to generate a finite set of admissable rules R_{L^a}, whereas the constraint (ii) on the remaining values entails that the *new* value v_k that forms L^{k+1} from L^k leads to smaller and smaller admissible sets: $R_{L^{k+1}} \subseteq R_{L^k}$. It is worth noting that $R_{L^{k+1}}$ plays the role of what the learning communities call the *version-space* [13], i.e. the set of hypotheses still plausible after having considered an input set of labeled values.

Example 2. *Consider again the above Example 1 and our running example in Figure 1. Then a possible t.s. is* $L^2 = \{v_0^+, v_1^+\}$ *and* $R_{L^2} = \{r_2, r_6, r_5\}$*. Possible candidate values are* $V_{L^2}^R(U) = \widehat{V}_{L^2}^R(U) \setminus L^2 = \{$'anthony.h.me', 'Oscar'$\}$*. A new* MQ *can be formed by choosing a new value* v_2 *to query from the elements in* $V_{L^2}^R(U)$*. E.g., "*is 'anthony.h.me' a correct value? *".*

In the following we will uniformly refer to both L and one of its observed subsets L^k blurring the differences between the two concepts whenever the context clarifies which one is actually involved.

It can always be decided whether a rule extracting the desired vector exists. However, since it is not known in advance whether that rule was in the set of all candidate rules, the only certain way to be sure of its presence is by checking every single page [1].

3. PROBABILISTIC MODEL TO EVALUATE WRAPPER QUALITY

We now introduce a probabilistic model for evaluating the quality of a wrapper expressed as an extraction rule r taken from a class of candidate extraction rules $\mathcal{R}$.

Our model computes: (i) the probability $P(r|L^{k+1})$ that a rule r is correct, observed a t.s. L^{k+1}; (ii) the probability $P(\overline{\mathcal{R}}|L^{k+1})$ that the correct rule is not in $\mathcal{R}$, observed L^{k+1}, i.e., that a correct rule has not been generated at all. Table 1 summarizes the notations used for the main events covered by our analysis, and their probabilities. By applying Bayes' theorem:

Table 1: The main events of the bayesian analysis

Probability Notation	*Event*
$\mathcal{P}(\mathcal{R})$ / $\mathcal{P}(\overline{\mathcal{R}})$	prior probability that a/none rule in $\mathcal{R}$ is correct
$P(v_k^l \mid r, L^k)$	likelihood of v_k^l if r is correct, observed L^k
$P(v_k^l \mid \overline{\mathcal{R}}, L^k)$	likelihood of v_k^l if the correct rule is not in R_{L^k}, observed L^k
$P(L^{k+1}) = P(v_k^{l_k}, L^k)$	probability of a t.s. L^{k+1}

$$P(r|L^{k+1}) = \frac{P(v_k^l|r, L^k)P(r|L^k)}{P(v_k^l|L^k)} \quad (1)$$

$$P(\overline{\mathcal{R}}|L^{k+1}) = \frac{P(v_k^l|\overline{\mathcal{R}}, L^k)P(\overline{\mathcal{R}}|L^k)}{P(v_k^l|L^k)} \quad (2)$$

where $P(v_k^l|L^k)$ is a *normalization factor* that can be expressed as:

$$\sum_{r \in \mathcal{R}_{L^k}} P(v_k^l|r_i, L^k)P(r_i|L^k) + P(v_k^l|\overline{\mathcal{R}}, L^k)P(\overline{\mathcal{R}}|L^k)$$

For any k, $P(r|L^{k+1})$ and $P(\overline{\mathcal{R}}|L^{k+1})$ can be defined iteratively by means of $P(v_k^l|r, L^k)$, $P(v_k^l|\overline{\mathcal{R}}, L^k)$, $P(\overline{\mathcal{R}}|L^k)$ and $P(r|L^k)$. The probabilities $P(v_k^l|r, L^k)$ and $P(v_k^l|\overline{\mathcal{R}}, L^k)$ can be defined by abstracting the actual process that leads to the observed t.s. into a simple *generative model*. This is essentially equivalent to define a p.d.f. over every t.s.

By repeatedly applying the bayesian updating rules expressed by equations 1 and 2, the model allows the computation of $P(\overline{\mathcal{R}}|L^{k+1})$ and $P(r|L^{k+1})$ for any k, starting from *prior*-probabilities $P(\mathcal{R}|L^a) = \mathcal{P}(\mathcal{R})$ of having generated a correct rule in $\mathcal{R}$, and the probability $P(r|\mathcal{R}, L^a)$ of r being a correct rule once the t.s. L^a has been observed. The iteration continues until admissible rules exist, i.e., until $\mathcal{R}_{L^k} \neq \emptyset$.

3.1 Bootstrapping Probabilities

For bootstrapping our probabilistic model, the following probabilities are needed: (i) the prior probability $\mathcal{P}(\mathcal{R})$ that a correct rule has been generated in the class of rules $\mathcal{R}$, and; (ii) the probability $P(r|\mathcal{R}, L^a)$ that the extraction rule $r \in \mathcal{R}$ does extract the correct vector of values from the input set of pages U once its has been observed the initial set of annotations L^a.

For the former prior $\mathcal{P}(\mathcal{R})$, we follow a standard approach, and estimate it by measuring the frequency of the involved events on a sufficiently large set of attributes: $\mathcal{P}(\mathcal{R})$ has been fixed to $\frac{n_h}{N}$ where n_h is the number of attributes extracted by a rule in $\mathcal{R}$ and N is the number of attributes sampled (we considered $N = 290$ attributes).

As regards the latter p.d.f. $P(r|\mathcal{R}, L^a)$, if $|R_{L^a}(U)| > 0$ we redistribute $\mathcal{P}(\mathcal{R})$ according to a uniform p.d.f. over all the vectors extracted by admissible rules $r \in R_{L^a}$, as follows:

$$P(r|\mathcal{R}, L^a) = \frac{n}{|R_{L^a}(U)|} \cdot \mathcal{P}(\mathcal{R});$$
where: $n = |\{r', r' \in R_{L^a}(U) : r'(U) = r(U)\}|$.

3.2 Generative Model for Training Sequences

We now describe the generative model for the training sequences: it is used to derive the posterior probability

$P(r|L^k)$ that $r(U)$ is the correct vector to extract once a t.s. L^k has been observed. This vector is not known in advance, but the values forming the t.s. L^k will be labeled as either positive or negative according to it.

Let $\mathcal{P}(\mathcal{R})$ be the prior probability that the correct vector can be extracted by a rule belonging to $\mathcal{R}$. We suppose that the acquisition of a new labeled value v_k to form L^{k+1} from L^k follows a uniform p.d.f. amongst all values still queryable, i.e., the values in $V^{\mathcal{R}}_{L^k}(U) = \widehat{V}^{\mathcal{R}}_{L^k}(U) \setminus L^k$. Similarly, given a correct rule r, let $V^+(L^k, r) = V^{\mathcal{R}}_{L^k}(U) \cap r(U)$ denote the set of all and only the values that can form new positive values, and $V^-(L^k, r) = V^{\mathcal{R}}_{L^k}(U) \setminus V^+(L^k, r)$ the set of values that can form negative values. It follows:

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

Similarly, we can compute $P(v^l_k|\overline{\mathcal{R}}, L^k)$ following an approach based on a uniform p.d.f. over all possible values. These are essentially all the values in $V^{\mathcal{R}}_{L^k}(U)$ but only the values in $V^{\mathcal{R}}_{L^k}(U) \cap \overline{\mathcal{R}}_{L^k}(U)$ can be labeled either positive or negative (and we assume with the same probability) while the values in $V^{\mathcal{R}}_{L^k}(U) \setminus \overline{\mathcal{R}}_{L^k}(U)$ will surely be labeled negative. Therefore, it follows that $P(v^l_k|\overline{\mathcal{R}}, L^k) =$

$$\begin{cases} P(v^+_k|\overline{\mathcal{R}}, L^k) = \frac{1}{2 \cdot |V^{\mathcal{R}}_{L^k}(U) \cap \overline{\mathcal{R}}_{L^k}(U)|}, \text{ iff } v_k \in V^{\mathcal{R}}_{L^k}(U) \cap \overline{\mathcal{R}}_{L^k}(U) \\ P(v^-_k|\overline{\mathcal{R}}, L^k) = \frac{1}{|V^{\mathcal{R}}_{L^k}(U) \setminus \overline{\mathcal{R}}_{L^k}(U)|} \text{ , iff } v_k \in V^{\mathcal{R}}_{L^k}(U) \setminus \overline{\mathcal{R}}_{L^k}(U) \\ 0 \text{ , iff } v_k \notin V^{\mathcal{R}}_{L^k}(U) \end{cases}$$

Note that the exact computation of the set $\overline{\mathcal{R}}_{L^k}(U)$ can be expensive, since given a value $v \in V^{\mathcal{R}}_{L^k}(U)$, in order to figure out whether $v \in \overline{\mathcal{R}}_{L^k}(U)$, we should enumerate a potentially very large number of vectors in $\overline{\mathcal{R}}_{L^k}(U)$.

We adopt an approximate and efficient solution based on the assumption that the equivalences holding for $k = 0$:[1] $\overline{\mathcal{R}}_{L^k}(U) = V^{\mathcal{R}}_{L^k}(U)$ and $V^{\mathcal{R}}_{L^k}(U) \setminus \overline{\mathcal{R}}_{L^k}(U) = \emptyset$, also hold for any $k > 0$. Hence, it can be rewritten as:

$$P(v^l_k|\overline{\mathcal{R}}, L^k) \simeq \begin{cases} \frac{1}{2 \cdot |V^{\mathcal{R}}_{L^k}(U)|} & \text{, iff } v_k \in V^{\mathcal{R}}_{L^k}(U) \\ 0 & \text{, iff } v_k \notin V^{\mathcal{R}}_{L^k}(U) \end{cases} \quad (3)$$

Actually, this is an oversimplification when k gets bigger and approaches $|U|$: both $\overline{\mathcal{R}}_{L^k}(U)$ and $V^{\mathcal{R}}_{L^k}(U)$ gets smaller and smaller and $V^{\mathcal{R}}_{L^k}(U) \setminus \overline{\mathcal{R}}_{L^k}(U) \neq \emptyset$. Since the algorithm looks for the correct rule while minimizing k, in our setting this semplification does not significantly affect the results.

4. LEARNING EXTRACTION RULES

The probabilistic model developed in the previous section aims at computing, observed a t.s. L^{k+1}, the probability $P(r|L^{k+1})$ that a given extraction rule r within a set of candidate rules $\mathcal{R}$ is correct, and the probability $P(\overline{\mathcal{R}}_{L^{k+1}})$ that the correct rule is not inside $\mathcal{R}$.

We now present an algorithm, called ALF, that exploits the model to infer a wrapper. ALF aims at inferring extraction rules with a high probability of correctness, while minimizing the length of the t.s., i.e., the number of membership queries to be posed to the crowd.

[1]Admitting that every value is extracted at least by one rule.

As we discussed in Section 1.1, the length of the t.s. depends on the expressiveness of the class of rules. Rather then relying on a statically designed (and possibly oversize) class of extraction rules, ALF organizes the class of candidate rules $\mathcal{R}$ into a hierarchy of classes $\{\mathcal{R}^h\}_{0 \leq h \leq m}$ of increasing expressiveness.

The algorithm starts by looking for a rule within the class $\mathcal{R}^0$ of the lowest expressiveness and computes the probability of its correctness. If such a probability is not satisfactory, the algorithm expands the class to the larger class $\mathcal{R}^1$, and consequently poses more membership queries, thus enlarging the t.s. In order to choose the appropriate membership queries, ALF uses an active approach by selecting the best queries to minimize its total number as further discussed in Section 4.1. The process is repeated until either it finds a rule of satisfactory probability, or it concludes that it is unlikely that this rule exists within the considered hierarchy.

The approach is independent of the details of the formalism used to express the extraction rules. In our implementation, we make use of XPath expressions; namely, we use *absolute* and *relative* XPath expressions. The former specify paths from the root to the leaf node that contains the value to be extracted; the latter are paths starting from a generic pivoting node.[2] Relative XPath expressions are classified based on the path length.[3] Our hierarchy $\{\mathcal{R}^h\}$ organizes absolute and relative XPath expressions as follows: $\mathcal{R}^0$ is the *class of absolute XPath expressions*, $\mathcal{R}^h$, for $0 < h \leq m$, is obtained by adding to $\mathcal{R}^{h-1}$ the class of relative XPath rules with path length h.

Listing 1 ALF: An active learning algorithms for extraction rules

Input: a set of pages $U = \{p_1, \ldots, p_{|U|}\}$
Input: a set of initial annotations $L^a = \{v^+_o, \ldots, v^+_{a-1}\}$

Parameter: a hierarchy of rules $\{\mathcal{R}^h\}$ over U

Output: $P(r|L^{k+1})$ over $r \in \mathcal{R}^h_{L^{k+1}}$, $P(\overline{\mathcal{R}}^h_{L^{k+1}})$;

```
1: let h ← 0;
2: let R ← R^h_{L^a};
3: while (R ≠ ∅ and not HALT(R, L^k)) do
4:    v_k ← CHOOSEQUESTION(R, L^k);
5:    l ← ORACLE(v_k);
6:    L^{k+1} ← L^k ∪ {v^l_k};
7:    R ← R^h_{L^{k+1}};
8:    compute P(r|L^{k+1}), ∀r ∈ R according to eq. 1;
9:    compute P(R̄^h|L^{k+1}) according to eq. 2;
10:   h ← h + EXPANDRULESET(R, L^{k+1});
11:   k ← k + 1;
12: end while
13: if (R ≠ ∅) then
14:    return R^h_{L^{k+1}}, P(r|L^{k+1}) and P(R̄^h|L^{k+1});
15: end if
16: return ⊥;
```

[2]In the current prototype only textual leaves are used as candidate pivot.

[3]The distance from the pivot to the extracted node is measured according to the number of edges crossed in the DOM representation of the HTML pages but considering contiguous siblings at distance 1.

Listing 1 contains the pseudo-code of the ALF algorithm: as input it takes a t.s. L built by actively [13] asking to an oracle (here modeled by means of the subprogram ORACLE()) the label of a value chosen by the subprogram CHOOSEQUESTION(); as output, it returns a p.d.f. describing the probability of correctness over the rules still admissible, and the possibility that a correct rules does not exist at all.

Initially, $\mathcal{R}^0$ is taken as initial set of candidate rules, and the set of rules admissible wrt the initial annotations $\mathcal{R}^0_{L^a}$ is computed (lines 1-2). In every iteration, the oracle is asked to label a new value v_k (lines 4-5) and the t.s. is updated to obtain L_{k+1} (line 6). Then, the set of admissible rules is updated (line 7) (recall that $R_{Lk+1} \subseteq R_{Lk}$), and the probabilities $P(r|L^{k+1})$ and $P(\overline{\mathcal{R}}_{Lk+1})$ are consequently updated (lines 8-9). EXPANDRULESET() has to decide whether the set of candidate rule should be expanded (line 10).

ALF can be instantiated by appropriately choosing the semantics of three subprograms: CHOOSEQUESTION(), which composes the next membership query, i.e., it selects the next value to be labeled by the user; HALT(), which establishes an exit criterion before the t.s. naturally expires (i.e., R becomes empty); EXPANDRULESET(), which decides at runtime whether $\mathcal{R}^h$ should be expanded with new candidate rules by incrementing h. The latter decision is based on the probability that the current class of rules cannot contain a correct rule ($P(\overline{\mathcal{R}}^h_{L^{k+1}})$): the higher its value, the more likely new candidate rules are needed, and thus the class of rules needs to be expanded.

We now describes several strategies to instantiate the three subprograms.

4.1 Asking the Right Questions

The CHOOSEQUESTION() procedure chooses the next membership query: it decides the next value to be labeled. We propose three alternative strategies: ENTROPY, GREEDY, and LUCKY, plus a baseline algorithm RANDOM.

RANDOM: It chooses a random admissible value:

CHOOSEQUESTION(R,L) { **return** a random $v \in V^R_L(U)$; }

and it serves as a baseline against other strategies.

ENTROPY: It bases the choice on the p.d.f. of the extracted values: a simple strategy is to choose the value on which rules most disagree, appropriately weighted according to their probability. This is equivalent to compute the *vote entropy* [13] for each $v \in R_{L^k}(U)$:

$$H(v) = -[P(v^+|L^k)\log P(v^+|L^k) + P(v^-|L^k)\log P(v^-|L^k)] \quad (4)$$

where: $P(v^+|L^k) = \sum_{r\in\{r\in R_{L^k}: r(p_v)=v\}} P(r|L^k)$;

and: $P(v^-|L^k) = \sum_{r\in\{r\in R_{L^k}: r(p_v)\neq v\}} P(r|L^k)$.

are the probabilities that v is either a value to extract or an incorrect value, respectively. The next value is that maximizing the vote entropy:

CHOOSEQUESTION(R,L) { **return** $\text{argmax}_{v\in V^R_L(U)} H(v)$; }

This choice essentially removes the most uncertain value.

GREEDY: The construction of the whole version-space is inefficient, since it requires to enumerate all possible t.s.. However, the version-space can be exploited to find the quickest t.s. confirming that a given rule is a solution. Let us call such a kind of sequences *confirming* t.s.: they aim more at deciding as quickly as possible that a given rule is a solution, rather than at finding which is the solution.

In every search step, GREEDY "elects" the most likely rule to play the role of the solution, and then it *greedily* builds a confirming t.s. wrt that conjecture. If, after a few labeled values, that rule is confuted and removed from the version-space, the whole process is repeated by formulating another conjecture around the most likely rule in the remaining version-space.

In this setting, the query is selected by greedily taking the value extracted by the supposedly "correct" rule from the page on which most other rules behaves differently: if that value is labeled positive as expected, the largest number of rules is removed from the version-space.

CHOOSEQUESTION(R,L) { **return** $r^*(p^*)$ }
where: $r^* = \text{argmax}_{r\in R_{L(U)}} P(r|L)$;
$p^* = \text{argmax}_{p\in U} |\{r(p) : r(p) \neq r^*(p)\}|$.

As the cost of this approach depends on the size of the version-space, it can be relevant in the early stages of the searching. The next variant delays its construction until the best rule emerges as significantly more likely than other candidates.

LUCKY: It is a hybrid of the former two approaches, and it works in two phases: first, it accumulates enough evidence of the correctness of a rule by using ENTROPY; then, it switches to GREEDY modality to confirm it. The switch is triggered by a fixed threshold λ_{r^*} on the probability of the most likely rule r^*.

This approach can be seen as a generalization of GREEDY: at the beginning it waits to observe enough evidence before allocating all its trust on the most likely rule.

EXAMPLE 3. *Reconsider the running example in Figure 1, and the t.s.* $L^1 = \{\text{john.s.me}^+\}$. *Suppose that* $\mathcal{P}(R) = 0.96$ *and that the probability is equally distributed among the set of candidate rules.*

$P(v_1^+|L^1)$ and $P(v_1^-|L^1)$ can be computed as follow:

v_1	$P(v_1^+\|L^1)$	$P(v_1^-\|L^1)$
anthony.h.me	0.24	$0.24 \cdot 3$
laura.w.me	$0.24 \cdot 2$	$0.24 \cdot 2$
Oscar	$0.24 \cdot 2$	$0.24 \cdot 2$
5500	0.24	$0.24 \cdot 3$
...	...	...

From $P(v_1^l|L^1)$ by using Eq. 4 the entropy $H(v)$ can be obtained as follows:

v_1	$H(v_1)$
anthony.h.me	$-0.24 \cdot log(0.24) - 0.72 \cdot log(0.72) = 0.58$
laura.w.me	$-0.48 \cdot log(0.48) - 0.48 \cdot log(0.48) = 0.70$
Oscar	$-0.48 \cdot log(0.48) - 0.48 \cdot log(0.48) = 0.70$
5500	$-0.24 \cdot log(0.24) - 0.72 \cdot log(0.72) = 0.58$
...	...

Hence, ENTROPY *can chooses* $v_1 =$ laura.w.me *or* $v_1 =$ Oscar *as the next value to query to get* $L^2 = L^1 \cup \{v_1\}$. *Note that* $P(r_i|L^1) = 0.24$, $i = 1, \ldots, 6$ *(rules extracting the same vector are indistinguishable) and the set of admissible rules after L^1 are equally probable. In this case,* GREEDY *would end up with a random selection of the most likely rule.*

4.2 SRM: Dynamically Expanding the Rule Set

EXPANDRULESET() is in charge of deciding whether and when expanding the set of candidate rules used from a hierarchy of classes $\mathcal{R}^h$. We refer to this technique as SRM, since it is inspired by the *Structural Risk Minimization* principle originally proposed by the statistical learning community [15, 14] as a tool for dealing with the problem of overfitting.

It makes use of the probability $P(\overline{\mathcal{R}}^h_{L^k})$ that the correct rule is not present in the current set of candidate rules $\mathcal{R}^h$ after observing as input a given t.s. L^k.

We use a simple implementation EXPANDRULESET() based on a predefined fixed threshold $\lambda_{\overline{\mathcal{R}}}$ over $P(\overline{\mathcal{R}}_{L^k})$:

```
EXPANDRULESET(R, L) {
  if (R = R^m) return 0; // max expansion reached
  if (P(R̄_L) > λ_R̄) return +1;
  else return 0;
}
```

The set of rules is therefore enlarged *lazily*, i.e., only whenever according to $P(\overline{R}_L)$ there is evidence that a correct rule is not amongst the currently available candidates.

4.3 Termination strategies

The implementation of HALT() depends on the overall goal of the search strategy. A simple approach considers a minimum threshold λ_r on the probability of the best rule:

$$\text{HALT}(R, L)\ \{\ \textbf{return}\ \ (\text{argmax}_{r \in R_L} P(r|L) > \lambda_r);\ \}$$

This function looks for the best rule r that suits the t.s. and terminates as soon as it finds a rule with probability higher then λ_r. It is an appropriate solution in many practical settings.

5. SAMPLING STRATEGIES

So far we considered feasible the application of our algorithm ALF to the whole set of input page. However, in many practical cases this assumption is unrealistic because of the number of pages (e.g., consider www.imdb.com, which provides more than $6 \cdot 10^6$ pages about actors). Finding a sampling set that is "cheaper" to work on, and yet it represents a larger population, is a traditional statistic problem. In this section we contextualize this issue in our setting and move to the related problem of *sampling* the input pages into a much smaller set of sample pages. The extraction rules can be evaluated on the sample set much more efficiently than on the whole set of pages; at the same time, a *representative* sample set must preserve the power of differentiating the rules by showing all their differences. However, the sample pages need to be carefully selected to be representative while, at the same time, minimizing their number.

These aspects are often neglected in the literature. Typically, sample pages are selected randomly, or they are collected following straightforward crawling strategies. While random samples could end up not representing the whole set of pages, crawling strategies can lead to the composition of biased samples. As an example, www.imdb.com exposes its content mainly in the form of *top-lists*, such as top-list movies, top-list actors and so on. A crawler following the links in these lists will inherently collect biased samples concentrated around "famous" instances.

We formulate the problem of finding a set $I \subset U$ such that $|I| \ll |U|$ yet I is *representative* (with respect to a given class of extraction rules R) of all the pages in U. The representativeness of a set of pages $I \subset U$ wrt a set of rules R can be formalized by introducing the *disagreement set* of two extraction rules.

Given a set of pages P, and a set of rules R, the disagreement set, denoted as $D^P(r_i, r_j)$, between two rules $r_i, r_j \in R$, 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 vector of 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 wrt a set of rules R if and only if:

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

In other terms, I is representative of U wrt to R if all the differences amongst the rules in R are also observable on I.

EXAMPLE 4. *Consider again our running example in Figure 1 and suppose that $I = \{p_s, p_w\}$, while $U = \{p_h, p_s, p_w\}$. I does not represent U since $D^U(r_2, r_5) = \{p_h\}$ whereas $D^I(r_2, r_5) = \emptyset$.*

Given the set of input pages U, and the class of rules R, 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.[4] Set covering is an NP-complete problem but actually we do not need to compute the optimal sample set: it suffices to estimate it by considering a small but not necessary minimal set of pages.

Listing 2 proposes PAGESAMPLER, a greedy sampling algorithm to extract a representative set of pages I wrt a class of rules R from a large set of input pages U in $O(|U| \cdot |R_{L^a}|)$ time and $O(|R_{L^a}|)$ space.

Listing 2 PAGESAMPLER: A greedy sampling strategy

Input: a set of pages U;
Input: a class of rules R;
Input: a set of initial annotations L^a;

Output: a set $I \subseteq U$ that is *representative* of U; wrt R

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

PAGESAMPLER processes the whole set of pages U (lines 3-8). It maintains a set of pages I, initially empty, that is

[4]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.

Site	Entity	$\|U\|$	$\|I_C\|$
www.imdb.com	Actor	$5 \cdot 10^5$	30
www.imdb.com	Movie	$5 \cdot 10^5$	42
www.allmusic.com	Band	$5 \cdot 10^5$	36
www.allmusic.com	Album	$5 \cdot 10^5$	29
www.nasdaq.com	Stock quote	$7 \cdot 10^3$	15

Table 2: Dataset 1

Strategy	$\#MQ$ SRM off	$\#MQ$ SRM on	$\%MQ$ Saved	Precision SRM on	Recall SRM on
RANDOM	379	190	50%	0.998	0.977
GREEDY	398	169	58%	0.998	0.983
LUCKY	196	132	33%	0.996	0.995
ENTROPY	205	116	44%	0.998	0.99

Table 3: Total number of MQ for Dataset 1

representative wrt 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 admissible rules R_{L^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.

EXAMPLE 5. *Consider the running example and suppose that* PAGESAMPLER *has already processed p_s and p_w, producing $I = \{p_s, p_w\}$. Let $R_{L^1} = \{r_2, r_5, r_6\}$ be the set of admissible rules wrt to $L^1 = v_0^+ = \{\text{john.s.me}\}$. The pages in I do not differentiate r_5 from r_2 and r_6: $r_2(I) = r_6(I) = r_5(I)$. However, when processing the next page p_h,* PAGESAMPLER *detects the different behavior of r_5 wrt other rules: $r_2(p_h) = r_6(p_h) \neq r_5(p_h)$, and then adds it to I.*

To clarify how PAGESAMPLER is related to the disagreement sets, consider that if $|R_{L^a}(I \cup \{p\})| > |R_{L^a}(I)|$ it follows that there exist at least two rules $r_i, r_j \in R_{L^a}$ such that $r_i(p) \neq r_j(p)$ and $D^{I \cup \{p\}}(r_i, r_j) \setminus D^I(r_i, r_j) = \{p\}$. Conversely, if $|R_{L^a}(I \cup \{p\})| = |R_{L^a}(I)|$ then it follows that $D^{I \cup \{p\}}(r_i, r_j) \setminus D^I(r_i, r_j) = \emptyset, \forall r_i, r_j \in R_{L^a}$. Therefore, PAGESAMPLER maintains the representativeness of I for the subset of U already processed by adding a page p to I if and only if p changes the disagreement sets of the rules.

6. EXPERIMENTS

In this section we describe the experiments conducted to evaluate our approach. Section 6.1 presents the results of the learning algorithm ALF. Our experiments mainly concentrate on the impact of the SRM technique, which dynamically expands the class of the extraction rules: the results show that, thanks to SRM, ALF always reduces the number of membership queries, without penalizing precision and recall of the generated rules. Section 6.2 illustrates the results of experiments to evaluate the effectiveness of the sampling strategy implemented by the PAGESAMPLER algorithm. Our experimental results show that a few dozens of pages selected by PAGESAMPLER are sufficient to represent large collections of 10^5 pages from real-life websites.

6.1 Learning with ALF

We considered two distinct datasets to evaluate the learning algorithm ALF.

The first dataset has been obtained by downloading pages from large websites related to specific domain entities, as shown in Table 2. We wrote ad-hoc crawling programs, and let them collect around $5 \cdot 10^5$ pages for each entity from www.imdb.com and www.allmusic.com, and all the available pages about stock quotes from www.nasdaq.com (around $7 \cdot 10^3$). For each entity we selected about 10 attributes, for a total of 40 attributes. We manually crafted a golden XPath rule for every attribute to extract its values. The (non-null) values extracted by the golden rules over the whole sets of pages were then used to compute and evaluate the precision and recall of the best rule inferred by our learning algorithm ALF. For each rule r generated by our algorithm wrt a golden rule r_g, we used the standard metrics of precision (P), and recall (R), as follows: $P = \frac{|r_g(U) \cap r(U)|}{|r(U)|}$; $R = \frac{|r_g(U) \cap r(U)|}{|r_g(U)|}$.

We run the PAGESAMPLER algorithm over these sample sets of pages to derive a representative sample for every domain (the sizes of the input sets, $|U|$, and of representative samples, $|I_C|$, are shown in Table 2). Over these sets we run the ALF algorithm to infer the extraction rules or the target attributes. In this experiment we set the probability threshold that governs the halt condition to 0.9, and the maximum expressiveness to $\mathcal{R}^5$.

We were mainly interested to evaluate the impact of the SRM technique used by ALF and the different strategies to choose the next membership query. Table 3 summarizes the results of the experiment. We report the number of membership queries ($\#MQ$) with and without the SRM technique for all the CHOOSEQUESTION() strategies. Observe that SRM almost halves $\#MQ$. The most efficient strategy is ENTROPY, which significantly outperforms the baseline, represented by RANDOM, and GREEDY. However, there is a small price to pay: without SRM, a perfect rule is found (precision and recall equal 1.0 not shown in the Table 3), whereas SRM introduces a loss lower than 1%. This is due to early decisions made by the SRM technique: sometimes it decides to bet on the current, and imprecise, set of candidate rules rather than expanding the class and searching inside larger classes.

Another experiment aimed at considering the behavior of ALF by using different CHOOSEQUESTION() strategies, wrt the size of the hypothesis space, which in our context corresponds to the number of admissible vectors after the initial annotations, i.e., $|R_{L^a}(I)|$. Intuitively, the size of the hypothesis space is a measure of the cost that any learning algorithm needs to pay to infer a rule.[5]

The plots in Figure 2 show the average number of membership queries vs size of the hypothesis space. Observe that the SRM technique (plot on the bottom) always reduces the number of queries needed by ENTROPY wrt the case in which it uses a fixed expressiveness. Also, note that when $|R_{L^a}(I)|$ is low, the differences in terms of $\#MQ$ are not apparent. On the contrary, when $|R_{L^a}(I)| \gg 5$, RANDOM performs worse than other strategies. ENTROPY and LUCKY outperform the other approaches and, as expected from an active learning algorithm [13], $\#MQ$ follows a logarithmic trend with respect to the size of the hypothesis space.

[5]This is strictly related to the *sample complexity* commonly used by the machine learning community, as the amount of training data to learn a concept [3].

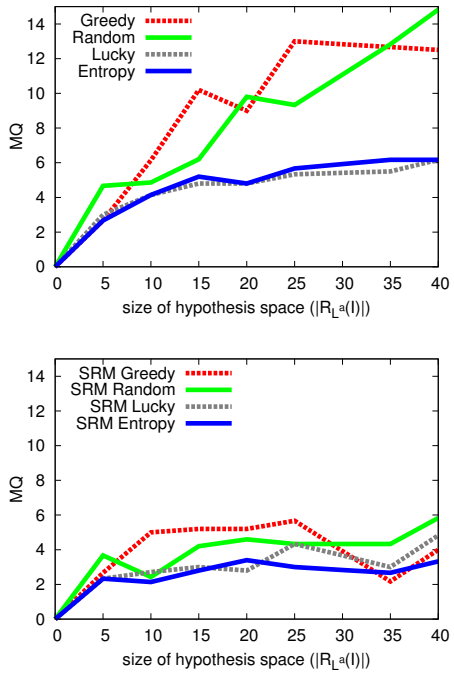


Figure 2: *#MQ* vs size of the hypothesis space with SRM disabled (top) and enabled (bottom)

Strategy	*#MQ* SRM off	*#MQ* SRM on	*%MQ* Saved
RANDOM	1092	728	34%
GREEDY	977	764	22%
LUCKY	903	684	25%
ENTROPY	880	683	25%

Table 4: Total number of *MQ* for Dataset 2

It is interesting to observe that GREEDY with SRM exhibits very good performances with attributes with a large hypothesis space, while it performs like RANDOM with SRM disabled. The explanation is that GREEDY concentrates the queries on the most likely hypothesis by building the whole version-space in order to find the shortest confirming t.s. However, our probabilistic model equally redistributes the uniform prior p.d.f. among all the admissible rules, and GREEDY ends up choosing, randomly, among a set of rules of the same probability. SRM solves this issue, as it does with RANDOM, by concentrating the random queries around the most likely class of rules.

To evaluate SRM in a different setting we used another dataset composed by a large number of attributes (250) from 100 websites, including popular ones, such as amazon.com, youtube.com, and ebay.com. Also for the attributes of this dataset, we manually wrote the golden rules. However, for each website we downloaded a small number of pages (a few dozens). As a consequence, we obtained a dataset with a large number of attributes, but with a limited hypothesis space. Even in this dataset, the application of SRM produces significant improvements, as reported in Table 4 (we do not report precision and recall, since we did not register any loss in this experiment).

Domain	Sampling	$\|I\|$	P	R
Movies	Crawler I_B	250	0.98	0.71
	Random I_R	250	0.99	0.99
	Representative I_C	42	1.00	1.00
Actors	Crawler I_B	250	1.00	1.00
	Random I_R	250	1.00	0.96
	Representative I_C	30	1.00	1.00
Stocks	Crawler I_B	86	1.00	0.98
	Random I_R	86	1.00	0.99
	Representative I_C	15	1.00	1.00
Albums	Crawler I_B	258	1.00	0.99
	Random I_R	258	1.00	1.00
	Representative I_C	29	1.00	1.00
Bands	Crawler I_B	289	1.00	0.68
	Random I_R	289	1.00	1.00
	Representative I_C	36	1.00	1.00

Table 5: Precision and recall with different sampling strategies

6.2 Sampling with PAGESAMPLER

We now discuss the experiments to evaluate the sampling algorithm PAGESAMPLER. For this evaluation we used the pages of the first dataset (Table 2). We collected three sample sets I according to different strategies, as follows:

- I_B represents a "biased sample": many large websites propose navigation paths to facilitate the browsing towards lists of relevant objects (e.g. famous actors, top-stocks, etc.). In our experiments, for each entity we downloaded the pages from the first list proposed by the sites. Therefore the size $|I_B|$ corresponds to the dimension of the proposed list.
- I_R is a set of pages randomly selected from the whole set U of pages with $|I_R|$ equals $|I_B|$.
- I_C is the representative sample set as computed by our sampling algorithm starting from the pages collected in our data set. $|I_C|$ is determined by the algorithm.

The first strategy does not pick up pages from the whole set of input pages U, while the second one chooses the sample pages in an uninformed way. These sampling strategies are used by many wrapper inference approaches more focused on the inference phase rather than on the sampling.

To evaluate the role of the three sampling strategies, Table 5 reports the average precision and recall computed over the attributes of all the entities of each domain. We inferred the extraction rules by running ALF (without SRM) on the three samples obtained. We obtained perfect rules when the inference was performed on the representative sample I_C; conversely, both the random set I_R and the biased set I_B loose precision and recall for a majority of cases, with a significant lost of recall with I_B for bands and movies.

Table 5 also reports the size of the samples: it is worth observing that the representative sample set I_C is always much smaller than the random sample set I_R. This is an important point as it affects the running times of the learning algorithm, which performs better when working on small samples. Figure 3 illustrates this issue: the graphic plots the average wrapper learning times (in logarithmic scale) vs the size of the random sample $|I_R|$ (number of pages). The curve associated to I_R describes the learning times to compute the wrapper using a random sample of increasing size. As an example, for a random sample of 50 pages, it runs in about 15

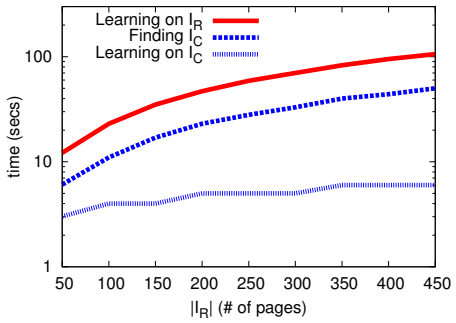


Figure 3: Wrapper learning times vs sample size

secs; for a random sample of 450 pages, it runs in 100 secs. The curve associated to the representative sample I_C reports the learning times to infer the wrapper over a representative sample I_C whose pages have been selected from a random sample I_R with that number of pages. For example, from a random sample of $|I_R| = 450$ pages, PageSampler selected a representative sample composed of $|I_C| = 25$ pages, and on this sample ALF inferred the wrapper in about 7 secs. Computing the representative sample has its own costs. However, as we can observe from the curves on Figure 3, even counting these costs the overall computation cost (sampling I_R to compute I_C + learning on I_C) is lower than the time required by learning without sampling (learning on I_R).

7. RELATED WORK

In machine learning, the number of labeled samples needed by a supervised learning algorithm to infer a *good* hypothesis is called *sample complexity* [3], and has been studied from several perspectives. For instance, similarly to our setting, [1] discusses 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?*". However, the main idea underlying our approach has been proposed by the statistical learning community [15], in which a *loss* function is given in order to characterize the quality of the produced hypothesis.

The *structural risk minimization* (SRM) [14], i.e., the decomposition of the set of hypotheses into a hierarchy of subclasses, aims at avoiding the *overfitting* problem: since the class of hypotheses studied by this community might be so expressive to be able to arbitrarily reduce the loss, a tradeoff with other quality criteria is needed to avoid that the learning algorithm selects the hypothesis perfectly describing the training data, rather than their underlying patterns.

Many researchers have proposed several variations of the learning paradigm to make it practically feasible in different applicative contexts: the learning approaches in which the inference algorithm is free to choose which sample to label next are usually defined *active* [13]. These have recently gained interest, since, as clarified in [3], they might produce exponential improvements over the number of samples wrt traditional supervised approaches.

To the best of our knowledge and differently from our proposal, all the approaches for inferring wrappers over structured websites developed by the researchers in the *wrapper inference* community [2, 4, 7, 11, 16], define the set of hypotheses *statically*, i.e., before performing the inference. Once set, the set of candidate rules cannot be changed without seriously revisiting the inference algorithm. Therefore they usually oversize the expressiveness of the formal language used to specify the extraction rules and additional samples are required only to compensate with the excess of expressiveness.

In this paper we concentrate on training data provided by means of a human intervention. A different solution to exploit supervised approaches without any human intervention consists of relying on existing repositories to automatically annotate web pages [8]. Unfortunately, in many domains suitable data does not exist at all (consider pages that publish subjective values, such as customer ratings, or real time data, such as stock quote prices). Also, the existing repositories might be biased over specific instances (typically, the most popular): such a biased information will annotate just a subset of the target pages, possibly preventing the generation of a valid wrapper.

Active learning approaches for wrapper induction have been proposed in [9, 12]. However, also in these works the expressiveness is *statically* defined. The latter approach requires complex user interaction, since the user has to choose the correct wrapper within a set of ranked proposals.

A few recent proposals try to scale the wrapper inference to the web scale [8, 10]. In [8] the authors leverage an available dataset, but they ignore the presence of biased samples (as suggested by its running example based on popular objects itself), while in [10] it is needed domain knowledge that only an human expert can provide.

8. CONCLUSIONS AND FUTURE WORK

Our work is mainly motivated by the success of crowd sourcing platforms, which can be used to scale wrapper generation. We propose a framework that allows supervised inference with simple mermbership queries suitable for the non-expert workers of a crowd platform. An original algorithm, ALF, applies active learning techniques to infer a wrapper, while minimizing the number of queries. ALF dynamically sets the expressiveness of the wrapper formalism, leading to a significant reduction of the number of queries needed to infer a wrapper. It does not depend on the specific classes of rules presented in the paper, and can be instantiated with other formalisms. We developed a complimentary sampling algorithm, PageSampler, to select for the learning phase a small yet representative set of sample pages from a much larger set of pages to wrap.

Experimental results prove the effectiveness of the approach. The dynamic expansion of the expressiveness of the wrapper formalism reduces the number of queries to learn a wrapper, with tangible cost savings. The sampling strategy leads to the selection of a small number of samples that effectively represents a much larger set of pages.

The quality model and the cost model proposed in our framework are the basis for future developments. For instance, the cost in term of total dollars spent for inferring a wrapper of the desidered quality can take into account both the cost of a PageSampler execution over large collection of pages on a cloud platform (such as EC2) and the number of *MQ* required by ALF on a crowd platform.

We are studying strategies to further optimize the interactions with the crowd. Namely, we are studying extensions of our bayesian model in order to manage worker's mistakes.

9. REFERENCES

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

[2] A. Arasu and H. Garcia-Molina. Extracting structured data from web pages. In *SIGMOD Conference*, pages 337–348. ACM, 2003.

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

[4] C.-H. Chang and S.-C. Lui. IEPAD: information extraction based on pattern discovery. In *WWW*, pages 681–688, 2001.

[5] R. Creo, V. Crescenzi, D. Qiu, and P. Merialdo. Minimizing the costs of the training data for learning web wrappers. In *VLDS*, pages 35–40, 2012.

[6] V. Crescenzi and G. Mecca. Automatic information extraction from large websites. *J. ACM*, 51(5):731–779, 2004.

[7] V. Crescenzi and P. Merialdo. Wrapper inference for ambiguous web pages. *Applied Artificial Intelligence*, 22(1&2):21–52, 2008.

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

[9] U. Irmak and T. Suel. Interactive wrapper generation with minimal user effort. In *WWW*, pages 553–563. ACM, 2006.

[10] T. Furche, G. Gottlob, G. Grasso, O. Gunes, X. Guo, A. Kravchenko, G. Orsi, C. Schallhart, A. J. Sellers, and C. Wang. DIADEM: domain-centric, intelligent, automated data extraction methodology. In *WWW (Companion Volume)*, pages 267–270. ACM, 2012.

[11] G. Gottlob, C. Koch, R. Baumgartner, M. Herzog, and S. Flesca. The lixto data extraction project - back and forth between theory and practice. In *PODS*, pages 1–12. ACM, 2004.

[12] I. Muslea, S. Minton, and C. A. Knoblock. Active learning with multiple views. *J. Artif. Intell. Res. (JAIR)*, 27:203–233, 2006.

[13] B. Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

[14] J. Shawe-Taylor, P. L. Bartlett, R. C. Williamson, and M. Anthony. Structural risk minimization over data-dependent hierarchies. *IEEE Transactions on Information Theory*, 44(5):1926–1940, 1998.

[15] V. Vapnik. An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10(5):988–999, 1999.

[16] Y. Zhai and B. Liu. Structured data extraction from the web based on partial tree alignment. *IEEE Trans. Knowl. Data Eng.*, 18(12):1614–1628, 2006.