ALFRED: Crowd Assisted Data Extraction

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. Crowdsourcing 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 demonstrate ALFRED, a wrapper inference system supervised by the workers of a crowdsourcing platform. Training data are labeled values generated by means of membership queries, the simplest form of queries, posed to the crowd. ALFRED includes several original features: it automatically selects a representative sample set from the input collection of pages; in order to minimize the wrapper inference costs, it dynamically sets the expressiveness of the wrapper formalism and it adopts an active learning algorithm to select the queries posed to the crowd; it is able to manage inaccurate answers that can be provided by the workers engaged by crowdsourcing platforms.

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, data extraction, 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. Several researchers investigated unsupervised data extraction solutions to scale the extraction task [3]. These solutions adopt sophisticated algorithms to exploit the regularity of web pages, but their applicability results compromised by the low precision of the inferred wrapper.

The recent advent of crowdsourcing 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, crowdsourcing 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 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 tasks, the number of training data produced by the crowd to infer a wrapper should be minimized.

In this work we present ALFRED [4], a supervised system that leverages workers engaged from a crowdsourcing platform to provide the needed training data. The tasks consist of a sequence of *membership queries* (*MQ*), which are the simplest form of queries, since they admit only a yes/no answer (e.g. "*Is 'City of God' the title of the movie in the page ?*") [1]. The *MQ* answers provided by the workers are exploited by ALFRED to infer the extraction rules.

To address the costs issue, our system proposes an *active* learning approach [7] that selects the most effective queries to quickly infer an accurate wrapper, thus minimizing the number and the cost of the tasks assigned to the crowdsourcing platform.

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

ALFRED addresses these issues: during the *sampling* phase the system processes the input pages to prune the large number of pages, selecting only a small yet representative set of pages; during the *learning* phase, the system infers the extraction rules by dynamically and lazily expanding the expressiveness of the formalism used to specify the extraction rules. Overall, the number of *MQ* needed is drastically reduced and the accuracy of the inferred rules is greatly increased, as explained in the following.

WWW 2013 Companion, May 13–17, 2013, Rio de Janeiro, Brazil.
ACM 978-1-4503-2038-2/13/05.

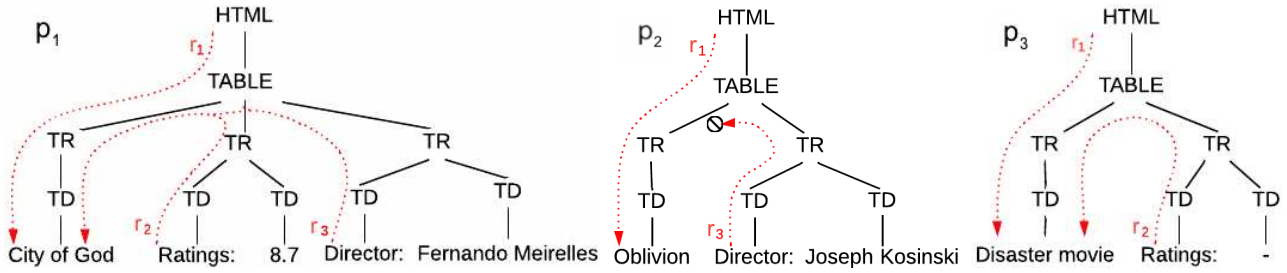

Figure 1: The DOM of three pages about movies

XPath rules	
r_1	/html/table/tr[1]/td/text()
r_2	//*[contains(.,"Rating:")]/../preceding-sibling::*[1]/td/text()
r_3	//*[contains(.,"Director:")]/../preceding-sibling::*[2]/td/text()

Table 1: XPath rules extracting the movie title

Sampling. Traditional approaches largely neglect the importance of correctly sampling large set of input pages. Most frequently, a fixed-size subset of randomly selected pages is chosen. However, this easily leads to *biased* samples that compromise the quality of the inferred wrapper, i.e., samples that do not show the full set of variations of the HTML template in the whole set of input pages. For example, a seldom optional field hardly shows in a small set of randomly selected pages.

Other approaches rely on locally available databases to automatically generate the needed training data [5, 6]. However, also the training data obtained in this way is often biased, since famous entities are much more frequently sampled than less popular ones.

With respect to these approaches, ours includes an exhaustive processing of the whole set of input pages which is reduced to a small but accurately selected unbiased set of representative pages.

Expressiveness. Consider the pages sketched in Figure 1: suppose that the Title of the movies has to be extracted. Observing the first annotated page p_1, a possible set of candidate rules are $\{r_1, r_2, r_3\}$ shown in Table 1. The correct rule r_1 can be separated from the wrong rules r_2 and r_3, by requiring additional training data, like pages p_2 or p_3 annotated with the correct value to extract. If the expressiveness of the extraction language is limited to simple rules like r_1, then a second sample page is not needed. On the contrary, if rules such as r_2 and r_3 are generated, additional training data are needed just to discard these wrong rules.

The formal languages for specifying wrappers in traditional approaches are designed statically, and their expressiveness cannot be changed without seriously revisiting the inference algorithm. Therefore, the class of extraction rules is usually oversize and additional samples are required only to compensate with the excess of expressiveness.

2. ALFRED ARCHITECTURE

The ALFRED approach to scale the data extraction issue consists of a supervised wrapper inference process, which relies on the results of mini tasks submitted to the crowd.

ALFRED takes as input a (possibly very large) collection of pages containing data of interest. The data to be extracted are specified by annotating its value over a single input page. Based on the input annotation, the system produces an initial set of candidate rules from which it has to select the most accurate one over the whole set of input pages. However, to keep the system efficient, the rules are evaluated over a small subset of the input pages suitably selected by a sampling algorithm. The system learns the rules by posing simple queries to crowd workers. To address the usual unreliability of crowdsourcing workers, the same tasks are proposed to several different workers, and the correct answers are determined based on a consensus based approach.

Figure 2 illustrates the architecture of the ALFRED system, as well as the flow of the process. The input of the system is a set of URLs and a first annotated sample page. The initial annotation represents an example of the data to be extracted; they can be produced manually, or they can be computed starting from data already available in a local repository.

The whole set of pages is then processed by the *Sampler*, which aims at selecting a representative set of pages, i.e., pages whose templates contain all the variants appearing in the whole input collection of pages. The representative set is then passed to the *Crowd Manager*, which generates multiple tasks and submits them to a crowdsourcing platform. The workers engaged by the crowdsourcing platform are redirected to an interactive web application interfaced to the *Active Learner* module.[1] Each worker is asked to accomplish a task, consisting of a set of membership queries. Each query asks to the worker whether a proposed value is correct or not. Based on the worker's answers, the *Active Learner* module produces the extraction rules. To manage the possible presence of inadvertent mistakes and adversarial "spamming" answers, the *Crowd Manager* submits a redundant number of tasks and solves the conflicts among the provided answers.

We now give a more detailed description of the main modules of the system.

2.1 Sampler

This module works off-line. It receives as input the whole set of input pages (e.g., the list of $2 \cdot 10^6$ movie pages from IMDB) and one annotated page. As output, the *Sampler* returns a smaller set of pages, which is representative of all the variations of the HTML template in the input set of

[1]A copy of the web application is available at `http://alfred.dia.uniroma3.it`.

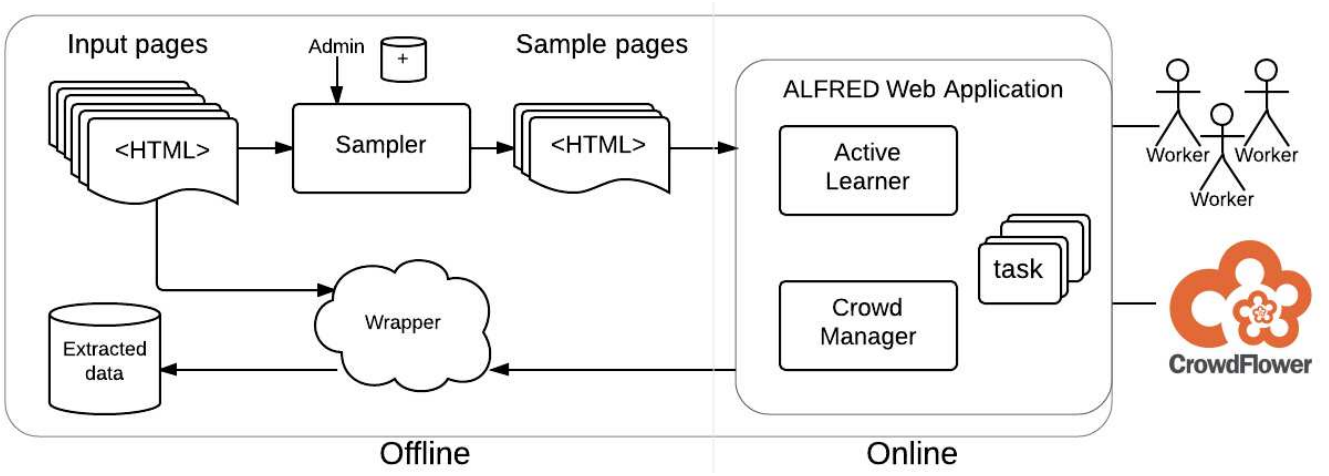

Figure 2: ALFRED **Architecture**

pages. Typically, even with very large collections, the output representative set contains just a few dozens of pages.

To detect the differences among the HTML templates, for each attribute the system automatically generates a first pool of candidate extraction rules that are capable of extracting its value over the annotated page. For example, suppose that we want to extract the title of movies from the pages in Figure 1, and that page p_1 is the initial page on which 'City of God' is the annotated value. The rules r_1, r_2, and r_3 shown in Table 1 represent examples of extraction rules that could be generated to extract such annotated value. We observe that for every pages that share the same template of p_1, the above three rules would behave in the same way, that is, they would extract the same value. However, these rules might return different values when applied on pages with a different template. Continuing our example, observe how r_1, r_2 and r_3 do not extract the same value when applied over p_2: r_1 extracts 'Oblivion', r_2 and r_3 do not extract anything.

Overall the different behaviors of candidate extraction rules make apparent differences among the HTML template of the input pages. Our sampling algorithm builds on these intuitions [4]. It works incrementally by selecting a subset of representative pages from the whole set of input pages: it adds a page to the representative sample whenever the page exhibits new differences among the candidate rules.

2.2 Active Learner

This module is at the core of the system: it implements an active learning algorithm that generates the queries to pose to the workers in order to infer the extraction rules while minimizing the learning costs [4].

As discussed in the previous section, the costs of learning an extraction rule depend on the expressiveness of the class of rules. Unlike traditional approaches we do not work with a statically defined class, but we adopt an original approach, inspired to *structural risk minimization* (SRM), a statistical learning principle [9, 8], 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; then, a class of rules is lazily expanded only if it is actually needed. To decide whether and when expanding the set of candidate rules, a probabilistic model evaluates the quality of the wrapper, by computing the probability that the candidate rules in the *current* class of rules are correct.

The algorithm starts by looking for a rule within the class with the lowest expressiveness, and computes the probability of its correctness. If such a probability is not adequate, the algorithm expands the class, and consequently poses more membership queries, thus enlarging the number of answers collected by the workers. In order to choose the appropriate membership queries, the algorithm uses an active approach, which aims at selecting the best queries to converge to the solution, then minimizing the total number of queries. The process is repeated until either it finds a rule with a satisfactory probability, or it concludes that it is unlikely that this rule exists.

2.3 Crowd Manager

This module manages the interaction with the crowdsourcing platform[2] with a twofold responsibility: it engages the workers from the crowdsourcing platform, and it solves possible conflicts among the workers' answers.

To enroll the workers, the *Crowd Manager* submits several tasks to the crowdsourcing platform. The engaged workers are redirected to the ALFRED web application, where they start to interact with the *Active Learner* module. Whenever a worker has answered to a sufficient number of queries to allow the system to infer a bunch of extraction rules, the *Crowd Manager* returns a code that the worker inserts into the crowdsourcing platform to prove the task fulfillment.

Presenting HTML pages downloaded from an external server into a web application under our control is not trivial: client side scripts and dependencies on remote resources could prevent the pages to be rendered outside the server originally publishing them. During the downloading, we generate pre-computed images of the pages selected by the *Sampler* to overcome these issues.

[2]We rely on CrowdFlower (`http://www.crowdflower.com`), a "meta" crowdsourcing platform interfaced to several crowdsourcing services, including Amazon Mechanical Turk.

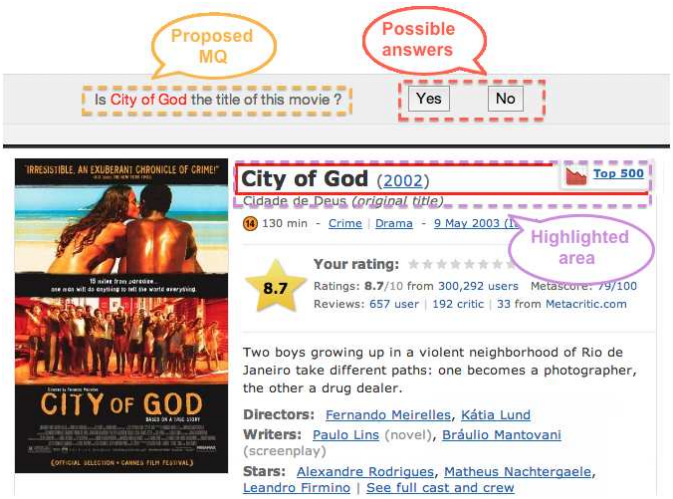

Figure 3: Web application interface

Site	Entity	# pages
www.imdb.com	Actor	$5 \cdot 10^6$
www.imdb.com	Movie	$2 \cdot 10^6$
www.allmusic.com	Band	$3 \cdot 10^6$
www.allmusic.com	Album	$2,4 \cdot 10^6$
www.nasdaq.com	Stock quote	$7 \cdot 10^3$

Table 2: Dataset

The *Crowd Manager* also deals with the intrinsic inaccuracy of the workers. We adopt a standard approach based on the submission of redundant tasks, and collect the answers to the membership queries assuming that they may include mistakes. Then, we apply an algorithm for learning from noisy samples [2] suitably adapted in our context. The algorithm ranks the set of candidate rules according to the number of conflicting answers; in our implementation, the answers are also weighted according to the estimated accuracy of the workers.

To estimate the accuracy of the workers, the *Crowd Manager* leverages some facilities provided by crowdsourcing platforms. First, it considers the general reputation of the worker; in addition, in order to check the reliability of the workers in the specific tasks proposed by ALFRED, the *Crowd Manager* submits to newly engaged workers a task composed by questions whose answers are know in advance to the system (a golden set). In this way, the *Crowd Manager* can directly evaluate the accuracy of the worker (or even reject bad workers).

3. DEMO DESCRIPTION

In this demonstration we present the different modules of ALFRED at work.[3] Table 2 shows the collections of pages that we use during the demonstration. Observe they contain a large number of pages. Since the *Sampler* needs a few hours to elaborate such a large number of pages, we pre-compute the representative sample sets: during the demonstration we show a log of the run, and the output sample pages.

We show how the *Sampler* is able to select a small representative set of a few dozens of pages from large input sets of millions of pages such as those in Table 2.

To illustrate the *Active Learner* module, we show the web application at work: the attendees act as workers of a crowdsourcing platform and they are asked to complete the tasks submitted by ALFRED. Each task consists of about 20 membership queries, which are posed by the web application through the interface shown in Figure 3: it visualizes one page, and proposes a simple question about one value extracted from the page. To ease the worker, the application also highlights the area containing the value involved in the query. Also, we demonstrate how the system works behind the scenes: we show the probability that it computes for the set of candidate extraction rules, and how it chooses the queries in order to quickly converge to the correct extraction rule. Finally, we demonstrate the SRM technique at work by showing the expansion of the current set of candidate rules operated by the *Active Learner* whenever the probability that the correct rule has been already generated becomes lower than a threshold.

The work done by the *Crowd Manager* is demonstrated by illustrating how it processes incorrect answers provided by the attendees. Also, in order to show the estimation of the workers' accuracy, we will present the statistics that we are collecting in several experiments that we are already running over real crowdsourcing platforms.

Finally, we submit several tasks on a crowdsourcing platform for demonstrating to the attendees the overall execution of our crowd assisted data extraction system.

[3]A screencast of the demonstration can be found in `http://www.youtube.com/watch?v=qEgGf-DQhq8`.

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

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

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

[6] 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.

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

[8] 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.

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