On Extracting Information from Semi-structured Deep Web Documents

Patricia Jiménez and Rafael Corchuelo(✉)

ETSI Informática, Avda. Reina Mercedes, s/n., 41012 Sevilla, Spain
{patriciajimenez,corchu}@us.es

Abstract. Some software agents need information that is provided by some web sites, which is difficult if they lack a query API. Information extractors are intended to extract the information of interest automatically and offer it in a structured format. Unfortunately, most of them rely on ad-hoc techniques, which make them fade away as the Web evolves. In this paper, we present a proposal that relies on an open catalogue of features that allows to adapt it easily; we have also devised an optimisation that allows it to be very efficient. Our experimental results prove that our proposal outperforms other state-of-the-art proposals.

Keywords: Information extraction · Semi-structured deep-web data sources

1 Introduction

Since the Web is currently the wealthiest source of data, many authors have worked on proposals whose goal is to help engineers create information extractors as automatically as possible.

Our focus is on learning a set of rules that can be used to extract the data of interest (positive examples) and discard the spurious data (negative examples) from semi-structured deep-web documents. There are many proposals that address this problem, but most of them are ad-hoc, that is, they rely on specific-purpose machine-learning techniques that were specifically tailored to the problem of extracting semi-structured web data [7,29,35]; many of them are even specific to a kind of layout, e.g., lists, tables, or search engine results [1,24]. This makes it difficult to adapt them as the Web evolves, since the features of the documents on which they rely and the techniques used to analyse them are built into the proposals. Consequently, extracting web data and structuring them has become quite an active research field for years, since existing proposals fade away quickly as the Web evolves. Only a few researchers have tried to use open catalogues of features to design their information extractors. Such open catalogues are appealing insofar they ease adapting the proposals as the Web evolves. Instead of devising a new algorithm to deal with the evolutions of the Web, one can focus on the features that capture the essence of such evolutions.

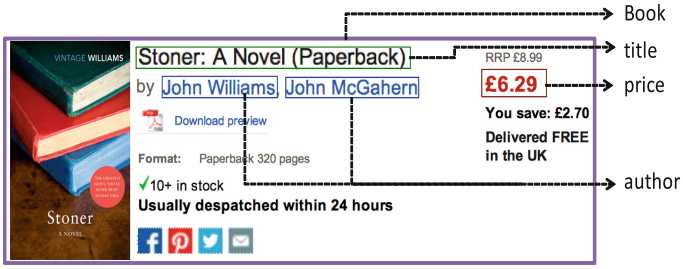


Fig. 1. Sample training document.

The few existing proposals in this category include SRV [13], Irmak and Suel's proposal [18], L-Wrappers [5], and Fernández-Villamor et al.'s proposal [12].

In this paper, we present a proposal to learn rules that are based on an open catalogue of attributive and relational features. By using relational features, our system is able to learn very expressive rules, since they consist of conditions that rely on first-order predicates with which we can model arbitrary properties of a web source. By using an open catalogue of features, our system can evolve as the Web does, which makes it very flexible and adaptable to changes. It follows a top-down covering approach, that is, it starts with the most general rule, and it iteratively adds conditions that are based on the features of the catalogue until the rule does not match any negative examples. The process finishes when every positive example is matched. Otherwise, the process continues to learn new rules. Our proposal also provides mechanisms not to produce very complex or too specific rules. To avoid the former, it includes a version of the Minimum Description Length principle; to avoid the latter, it tries not to include the most promising conditions only, but also conditions that can spread the search space and produce good rules in the forthcoming steps. Our proposal has to search through a typically large search space and the cost of evaluating candidate conditions is high if there are many positive and negative examples. To tackle this problem, we have incorporated a simple technique to discard some negative examples that has proven to work very well in practice.

The rest of the paper is organised as follows: Sect. 2 describes our proposal; Sect. 3 reports on our experimental analysis; Sect. 4 discusses on the related work; finally, Sect. 5 concludes the paper.

2 Description of Our Proposal

2.1 Training Sets

Our proposal works on a training set that is composed of positive examples, negative examples, and feature instantiations. To assemble it, we recommend downloading at least six documents that account for as much variability as possible, e.g., permuted or missing attributes, alternate formats, and so on.

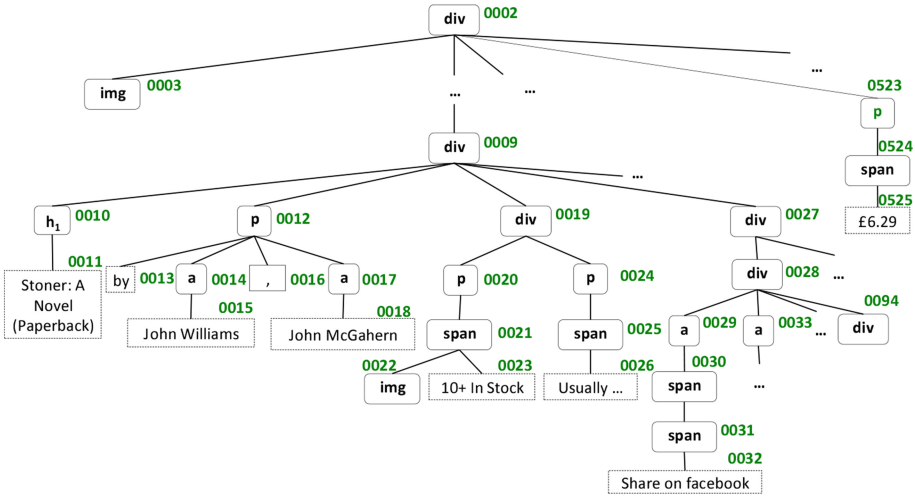


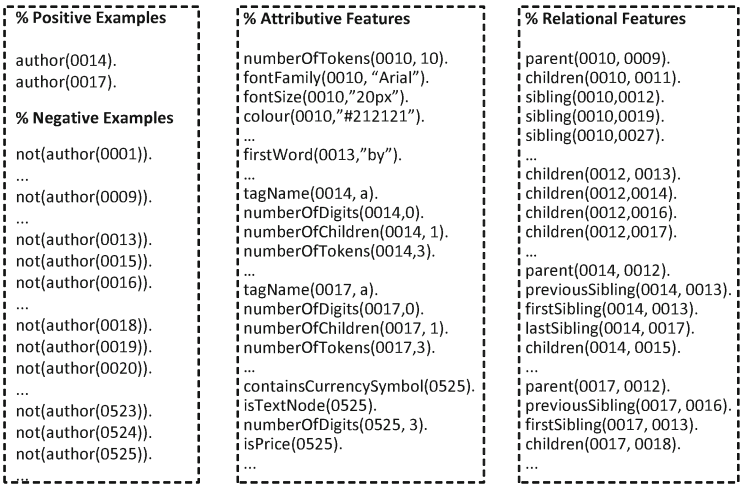
Fig. 2. A sample DOM tree.

Table 1. Partial catalogue of features. (W = word, N = DOM node, V = value.)

FEATURE NAME	FEATURE NAME	FEATURE NAME	FEATURE NAME
firstBigram(N,W,W)	numberOfBlanks(N,V)	numberOfIntegers(N,V)	endsWithPunctuation(N)
parent(N,N)	numberOfUppercaseBigrams(N,V)	isURL(N)	isAlphaNum(N)
lastSibling(N,N)	height(N,V)	width(N,V)	isNumber(N)
coordinates(N,V,V)	isUppercase(N)	backgroundColor(N,V)	numberOfUppercaseTokens(N,V)
firstSibling(N,N)	isCurrency(N)	isLowerCase(N)	isEmail(N)
tagName(N,V)	numberOfAlphaNum(N,V)	beginsWithPunctuation(N)	lineHeight(N,V)
children(N,N)	numberOfLowercaseTrigrams(N,V)	isCapitalised(N)	isBlank(N)
lastBigram(N,W,W)	numberOfChildren(N,V)	fontSize(N,V)	endsWithParenthesis(N)
firstWord(N,W)	numberOfSiblings(N,V)	isDate(N)	containsCurrencySymbol(N)
siblingIndex(N,V)	isNumber(N)	isbnNode(N)	verticalAlign(N,V)
lastWord(N,W)	containsBracketedNumber(N)	beginsWithParenthesis(N)	textAlign(N,V)
beginsWithNumber(N)	lowerCaseTokenAmount(N,V)	display(N,V)	borderBottomWidth(N,V)
nextSibling(N,N)	numberOfLetters(N,V)	fontWeight(N,V)	borderLeftWidth(N,V)
textLength(N,V)	numberOfCapital(N,V)	containsBracketedAlphaNum(N)	containsQuestionMark(N)
textNode(N)	numberOfDigits(N,V)	isYear(N)	isPhone(N)
depth(N,V)	numberOfUpperCaseTrigrams(N,V)	numberOfLowercaseBigrams(N,V)	...
numberOfTokens(N,V)	textDecoration(N,V)	marginBottom(N,V)	
sibling(N,N)	hasNotBlanks(N)	borderRightColor(N,V)	

Figure 1 shows a very simple document that a user has annotated, that is, he or she has specified which the positive examples are and has labelled them; Fig. 2 shows its corresponding DOM tree view. This document must be transformed into a training set by computing the features of every node. Table 1 shows a partial view of our open catalogue, which includes lexical, HTML, rendering or semantic features, to mention a few categories, and Table 2 shows an excerpt of our sample training set.

Table 2. An excerpt of a training set.



```
% Positive Examples

author(0014).
author(0017).

% Negative Examples

not(author(0001)).
...
not(author(0009)).
...
not(author(0013)).
not(author(0015)).
not(author(0016)).
...
not(author(0018)).
not(author(0019)).
not(author(0020)).
...
not(author(0523)).
not(author(0524)).
not(author(0525)).
```

```
% Attributive Features

numberOfTokens(0010, 10).
fontFamily(0010, "Arial").
fontSize(0010,"20px").
colour(0010,"#212121").
...
firstWord(0013,"by").
...
tagName(0014, a).
numberOfDigits(0014,0).
numberOfChildren(0014, 1).
numberOfTokens(0014,3).
...
tagName(0017, a).
numberOfDigits(0017,0).
numberOfChildren(0017, 1).
numberOfTokens(0017,3).
...
containsCurrencySymbol(0525).
isTextNode(0525).
numberOfDigits(0525, 3).
isPrice(0525).
...
```

```
% Relational Features

parent(0010, 0009).
children(0010, 0011).
sibling(0010,0012).
sibling(0010,0019).
sibling(0010,0027).
...
children(0012, 0013).
children(0012,0014).
children(0012,0016).
children(0012,0017).
...
parent(0014, 0012).
previousSibling(0014, 0013).
firstSibling(0014, 0013).
lastSibling(0014, 0017).
children(0014, 0015).
...
parent(0017, 0012).
previousSibling(0017, 0016).
firstSibling(0017, 0013).
children(0017, 0018).
...
```

```
1: procedure main(docs)
2:     result ← ∅
3:     for each different label in the annotations of docs do
4:         trainingSet ← CreateTrainingSet(docs, label)
5:         trainingSet ← ReduceNegatives(trainingSet)
6:         ruleset ← LearnRuleSet(label, trainingSet)
7:         result ← result ∪ {label ↦ ruleSet}
8:     end for
9:     return result
10: end procedure
```

Fig. 3. The main procedure.

2.2 The Main Procedure

Figure 3 shows our main procedure. It works on a collection of documents and iterates through the labels in the user annotations. In each iteration, it creates a training set and pre-processes it in order to reduce the number of negative examples, which speeds up the learning process since there are typically many such examples. By repeated experimentation we found that the best alternative was to remove 40 % of the negative examples randomly. Finally, this procedure returns a map in which each of the labels in the input documents is associated with a rule set that is specifically tailored to identifying positive examples of the corresponding types.

Learning a Rule Set: Figure 4 shows our procedure to learn a rule set. It works on a training set and a label, and it returns a set of rules. It first creates an empty set of rules and proceeds to create a single rule in each iteration.

```
1: procedure LearnRuleSet(label, trainingSet)
2:     ruleSet ← ∅
3:     repeat
4:         rule ← LearnRule(label, trainingSet)
5:         if rule ≠ nil then
6:             (ruleSet, trainingSet) ← Update(ruleSet, trainingSet, rule)
7:         end if
8:     until rule = nil ∨ trainingSet has no positive examples
9:     ruleSet ← PostProcess(ruleSet)
10:    return ruleSet
11: end procedure
```

Fig. 4. Procedure to learn a rule set.

If it succeeds, then the rule set is updated with the new rule, and the training set is subtracted the positive examples that it matches. The loop finishes when no new rule can be learnt or no positive examples remain in the training set. Finally, the rule set is post-processed to remove subsumed rules and useless conditions.

Learning a Rule: Figure 5 presents the procedure to learn a rule. It works on a label and a training set, and it starts with an empty rule. Then, it generates a set of possible refinements. Each refinement is a condition that consists in an instantiated feature or a built-in comparator. A refinement of the first type is a feature defined in the catalogue in Fig. 1, but the arguments are instantiated with variables of the corresponding types. The variables can be new variables that expand the search space or bound variables that have been introduced in the head of the rule or in previous conditions of the body of the rule. The only constraint is that there must exist a bound variable in every possible refinement. A built-in comparator is a binary condition that compares two variables or a variable and a constant, as long as they both are of the same type. The types with which a built-in comparator works can be numeric or discrete. We use five built-in comparators, namely: =, >, >=, <, and <=. When the variables to compare are discrete, only the comparator = is available.

In each iteration it attempts to refine the rule by adding conditions, so that it matches as many positive examples as possible and as few negative examples as possible. To do that, it implements an outer loop that iterates until no more refinements can be added or until the rule being learnt is a solution or too complex; in every iteration of the inner loop, it first computes a set of possible refinements for the current rule, that is, a set of conditions that might be added to the rule; then it evaluates each condition, that is, it computes a score that allows to compare its goodness to others, and it checks whether it is bad enough to be pruned immediately. Refinements that are not pruned are stored in a candidate set together with their scores. A candidate rule is the result of adding a candidate refinement to the current rule. The inner loop is executed as long as the current rule can be refined and it is not a solution; a rule is a solution when it matches some remaining positive examples and no negative ones.

```
1: procedure LearnRule(label, trainingSet)
2:     rule ← CreateEmptyRule(label)
3:     repeat
4:         stop ← false
5:         candidates ← ∅
6:         refinements ← GenerateRefinements(rule)
7:         for each refinement in refinements while ¬stop do
8:             (score, pruned) ← Evaluate(rule, refinement, trainingSet)
9:             if ¬pruned then
10:                candidate ← (refinement, score)
11:                candidates ← candidates ∪ {candidate}
12:                stop ← IsSolution(candidate, trainingSet)
13:            end if
14:        end for
15:        if candidates = ∅ then
16:            rule ← nil
17:        else
18:            bestCandidates ← FindBestCandidates(rule, candidates)
19:            rule ← add refinements in bestCandidates to rule
20:        end if
21:    until rule = nil ∨ IsSolution(rule, trainingSet) ∨ IsTooComplex(rule, trainingSet)
22:    if IsTooComplex(rule, trainingSet) then
23:        rule ← nil
24:    end if
25:    return rule
26: end procedure
```

Fig. 5. Procedure to learn a single rule.

Once the set of candidates is computed, the best ones are selected and added to the rule being constructed. When the outer loop finishes, the algorithm checks if the candidate rule is too complex, in which case, it returns *nil* to indicate that no good rule was found.

2.3 Output Rules

Next, we illustrate the rules that our proposal learnt to extract the data in Fig. 1. First, we learnt the following rule to identify book records:

$$\begin{aligned} book(X) \text{ :- } & children(X, Y), not(firstSibling(Y, Z)), \\ & tagName(Y, V), V = \text{“img”}, \\ & containsBracketedNumber(X), \\ & containsCurrencySymbol(X). \end{aligned}$$

It means that a book is a node whose first child is an image node and that some of its descendants has a number in brackets and a currency symbol.

Table 3. Labelled information on the first group of datasets

CATEGORY	LABELLED INFORMATION
Books	Book (record), title, author, price, isbn, and year
Movies	Movie (record), title, director, actor, year, and runtime
Cars	Car (record), make, color, doors, engine, mileage, model, price, transmission, type, and location
Conferences	Conference (record), date, place, title, and url
Doctors	Doctor (record), name, address, phone, fax, and specialty
Jobs	Offer (record), company, location, and category
Real Estate	Property (record), address, bedrooms, bathrooms, size, and price
Sports	Player (record), name, birth, height, weight, club, country, position, age, and college

The rule to identify the titles is as follows:

$$title(X) :- not(firstSibling(X, Y)), \\ tagName(X, Z), Z = \text{“h1”}.$$

It means that a title is a node that does not have a left sibling and is within an H1 header.

The rule for the author label is as follows:

$$author(X) :- tagName(X, Y), Y = \text{“a”}, \\ firstSibling(X, Z), \\ firstWord(Z, V), V = \text{“by”}.$$

It means that an author is a hyper-link node whose first sibling starts with the word “by”.

Finally, the rule for the price label is as follows:

$$price(X) :- isPrice(X), fontWeight(X, Y), \\ Y = \text{“bold”}, fontSize(X, Z), Z = \text{“18.7px”}.$$

It means that a price is an element node whose text matches a price format, whose font weight is “bold”, and whose font size is “18.7px”.

3 Experimental Analysis

We used a computer that was equipped with an Intel Core i7-2600 that ran at 3.34 GHz, had 8 GiB of RAM, Windows 7 Pro 64-bit, Oracle’s Java Development Kit 1.7.0_09, and SWI-Prolog 6.2.3. We used a collection of 25 datasets that provides a total of 657 documents. We gathered them from 20 real-world web sites and the remaining were downloaded from the RoadRunner and the RISE public repositories. For the first group of datasets, we downloaded 30 documents from each web site and handcrafted a set of annotations with the data that we wished to extract from each document, namely, data on books, cars, conferences, doctors, jobs, movies, real estates, and sport players. The information labelled from these web sites is shown in Table 3. The second group contains some of the datasets available at the RoadRunner and the RISE repository that provide

Table 4. Experimental results.

	Trinity			RoadRunner			FivaTech			SoftMealy			WIEN			OUR PROPOSAL		
Summary	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
Mean	0.968	0.968	0.966	0.575	0.567	0.571	0.829	0.882	0.848	0.867	0.691	0.724	0.694	0.617	0.627	0.985	0.986	0.984
Standard deviation	0.065	0.065	0.056	0.399	0.396	0.398	0.168	0.110	0.134	0.103	0.312	0.285	0.254	0.322	0.301	0.026	0.023	0.020
Dataset	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
Abe Books	1.000	1.000	1.000	0.648	0.585	0.615	0.917	0.991	0.953	0.873	0.578	0.696	0.516	0.160	0.245	1.000	1.000	1.000
Better World Books	0.989	1.000	0.994	-	-	-	0.989	0.958	0.973	0.979	0.985	0.982	0.428	0.350	0.385	1.000	1.000	1.000
Waterstones	0.961	1.000	0.980	0.978	0.867	0.919	1.000	0.944	0.971	1.000	1.000	1.000	0.714	0.675	0.694	0.992	1.000	0.996
IMDB	0.928	0.861	0.893	0.393	0.353	0.372	-	-	-	0.878	0.847	0.862	0.376	0.376	0.376	0.940	0.993	0.961
Disney Movies	1.000	1.000	1.000	0.667	0.667	0.667	0.712	0.671	0.691	0.970	0.970	0.970	0.718	0.718	0.718	0.984	0.992	0.988
All Movies	0.971	0.956	0.963	0.267	0.259	0.263	0.794	0.740	0.767	0.930	0.288	0.439	0.131	0.072	0.093	1.000	0.975	0.987
Auto Trader	0.992	1.000	0.996	-	-	-	-	-	-	0.889	0.870	0.880	0.889	0.000	0.000	1.000	0.981	0.990
Car Max	1.000	1.000	1.000	0.981	0.981	0.981	0.449	0.889	0.597	0.892	0.892	0.892	0.875	0.875	0.875	1.000	1.000	1.000
Classic Cars for Sale	0.859	0.904	0.881	0.493	0.493	0.493	-	-	-	0.896	0.896	0.896	0.170	0.128	0.146	0.991	0.967	0.978
All Conferences	0.984	0.992	0.988	0.722	0.722	0.722	0.840	0.900	0.869	0.993	0.253	0.404	0.800	0.400	0.533	0.939	0.992	0.961
Mbendi	1.000	1.000	1.000	0.867	0.867	0.867	0.903	1.000	0.949	0.600	0.600	0.600	0.800	0.400	0.533	1.000	1.000	1.000
Web MD	1.000	1.000	1.000	0.063	0.065	0.064	0.775	1.000	0.873	0.865	0.452	0.593	0.600	0.600	0.600	0.915	0.942	0.928
Steady Health	1.000	1.000	1.000	1.000	1.000	1.000	0.833	0.833	0.833	0.750	0.250	0.375	0.750	0.750	0.750	1.000	1.000	1.000
6 Figure Jobs	1.000	1.000	1.000	0.522	0.522	0.522	1.000	0.978	0.989	0.750	0.000	0.000	0.250	0.250	0.250	1.000	0.906	0.949
Career Builder	1.000	1.000	1.000	0.000	0.000	0.000	0.802	0.833	0.818	0.750	0.000	0.000	0.750	0.750	0.750	0.935	0.990	0.959
Homes	0.993	0.993	0.993	1.000	1.000	1.000	-	-	-	0.800	0.789	0.794	0.917	0.917	0.917	1.000	1.000	1.000
Remax	0.700	0.980	0.817	0.000	0.000	0.000	-	-	-	0.844	0.844	0.844	1.000	1.000	1.000	0.986	0.972	0.979
UEFA	1.000	1.000	1.000	1.000	1.000	1.000	-	-	-	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
ATP World Tour	0.971	0.994	0.982	0.967	0.967	0.967	0.990	0.878	0.930	0.943	0.943	0.943	0.714	0.714	0.714	0.949	0.970	0.958
NFL	1.000	1.000	1.000	0.978	0.978	0.978	0.530	0.813	0.642	0.714	0.714	0.714	0.857	0.857	0.857	0.994	1.000	0.997
Amazon (cars)	0.930	0.730	0.818	0.000	0.000	0.000	0.600	0.670	0.633	0.980	1.000	0.990	0.970	1.000	0.985	1.000	1.000	1.000
UEFA (teams)	0.990	0.990	0.990	0.917	0.917	0.917	0.970	0.990	0.980	0.810	0.890	0.848	0.640	0.750	0.691	1.000	1.000	1.000
Netflix	0.990	0.990	0.990	0.761	0.795	0.778	0.820	0.800	0.810	0.940	0.820	0.876	0.990	0.990	0.990	0.990	0.967	0.978
Bigbook	0.950	0.940	0.945	0.009	0.000	0.001	-	-	-	0.810	0.770	0.789	0.680	0.980	0.803	1.000	1.000	1.000
Zagat	1.000	0.860	0.925	0.000	0.000	0.000	1.000	0.980	0.990	0.810	0.630	0.709	0.810	0.720	0.762	1.000	1.000	1.000

semi-structured documents. From every web site, we selected six documents for training purposes and the remaining ones were used for validation purposes. We measured three effectiveness measures and two efficiency measures. The effectiveness measures are precision (P), which is the ratio of true positives of a rule with regard to the total number of true positives and false positives, that is, $P = tp/(tp + fp)$; recall (R), which is the ratio of true positives of a rule with regard to the total number of true positives and false negatives, that is, $R = tp/(tp + fn)$; and the $F1$ measure, which is the harmonic mean of precision and recall, that is, $F1 = 2\,P\,R/(P + R)$. The efficiency measures are the learning time (LT) and the extraction time (ET) as measured in CPU minutes; by extraction time, we mean the time required to execute a rule so that the data of interest is extracted from a web document. Since these measures are sensitive to unexpected experimentation conditions, we repeated the experiments 25 times and averaged the results after discarding some outliers using the well-known Cantelli's inequality.

We searched the Web and contacted many authors in order to have access to the implementation of as many proposals as possible. We managed to find an implementation for SoftMealy [17] and Wien [21], which are classical proposals, and RoadRunner [11], FivaTech [19], and Trinity [32], which are recent proposals. Results are presented in Table 4. A dash in a cell means that the corresponding proposal was not able to learn a rule for a given dataset. From this table,

we can confirm that our proposal clearly achieves the best effectiveness. It achieved an average precision of 0.985 ± 0.026, an average recall of 0.986 ± 0.023, and an average F_1 of 0.984 ± 0.020. Only Trinity can achieve results that are comparable to ours, although we outperform it: our precision was 0.016 ± 0.091 higher, our recall 0.018 ± 0.088 higher, and our F_1 0.018 ± 0.076 higher. Furthermore, the standard deviation of these measures is smaller, which means that our approach is generally more stable than the others, that is, it does not generally produce rules whose effectiveness largely deviates from the average.

4 Related Work

The literature provides many rule-based proposals. They build on a generic algorithm that interprets rules that are specific to a web site. These rules range from regular expressions to context-free grammars, Horn clauses, tree templates, or transducers, to mention a few. They can be handcrafted [14,26], which is a tedious and error-prone approach, learnt supervisedly [5,6,8,9,12,13,15–18,20,21,25,31,33], which requires the user to provide a training set in which he or she has annotated the positive examples, i.e., the data to extract), or unsupervisedly [2–4,10,11,19,22,23,27,28,30,34,36–38], which does not require an annotated training set, but a person to interpret the results.

In spite of the fact that learning such rules has historically been considered a text classification problem, very few proposals have explored using features of the tokens or the DOM nodes, e.g., their length, their colour, their depth, the ratio of letters, and the like. Neither have many proposals explored the idea of using relational features that allow to establish relationships amongst the examples. One of the main problems with exploring such relational features is that they cannot rely on a vector-based representation of the examples since the description of an example may involve features that are also computed from its neighbours.

Only a few proposals have explored the previous idea, namely: (a) SRV [13] starts with an overly-general rule and uses a specialisation procedure that is guided by the Information Gain function to learn a set of rules that can classify sequences of tokens; (b) Irmak and Suel's proposal [18] is an active learning approach, i.e., it learns several XPath-based rules that identify a number of records and then relies on a user to select the most appropriate ones; the procedure is repeated until the user is satisfied. (c) Bădică et al. [5] suggested to transform the input documents into a knowledge base and then use FOIL to infer rules that allow to characterise the DOM nodes that contain positive examples by means of their tags and the tags of their neighbours; (d) Fernández-Villamor et al.'s proposal [12] starts with a set of overly-specific rules and generalises them by combining pairs of rules as long as the overall F_1 measure improves.

Our proposal differs in that it relies on an extensive open catalogue of attributive and relational features whereas others hardly include a few attributive features and seldom use rendering features, which have proven to be necessary to

deal with current web documents. Additionally, none of these most-related proposals allow for recursion or negated conditions; ours does, which helps learn richer and more expressive rules.

Regarding computing negative examples, SRV works on text fragments, which makes it problematic when computing negative examples since the number of negative fragments is enormous. The authors included a hard bias so as not to enumerate the whole set of negative examples. In Irmak and Suel's proposal, the user guides the search for rules because negative examples cannot be generated. In L-Wrappers [5] negative examples are computed by using the well-known Closed-World Assumption as it is implemented by the FOIL system. In practice, the authors had to reduce the number of negative examples to roughly 0.1 % for the approach to be manageable; Fernández-Villamor et al.'s and our proposal work on DOM trees and negative examples are computed as the nodes that were not annotated by the user.

Regarding the learning algorithm, SRV is more inefficient since when a token in the neighbourhood helps discern well between positive and negative examples, it is bound by a relational condition and then the system explores only one of its attributive features. Thus, a discriminatory token in the neighbourhood has to be re-bound as many times as attributive features are necessary to be explored. This worsens the efficiency of the learning process. L-Wrappers is very inefficient in practice since the authors mention that extracting records with three or more attributes is infeasible in practice; they resorted to a technique that learns to extract pairs of attributes and then combine the results.

Only L-Wrappers and our proposal include mechanisms to limit the complexity of the rules, which is a means to avoid producing very-specific rules, to prune conditions that are not promising enough, which speeds up the learning process, and to recover from bad decisions by performing backtracking, which reduces significatively the chances to produce poor-quality rules.

5 Conclusions and Future Work

In this paper, we present a very effective proposal to learn rules that allows to extract the information of interest from deep-web sources automatically, so that it can be further processed by software agents. Since the search cost was high, we devised a technique to reduce the number of negative examples that turned it into a practical approach. Our results prove that our proposal is more effective than others in the literature, but, contrarily to them it can be easily evolved since it relies on an open catalogue of features. In future, we plan on analysing a series of heuristics to speed up the learning process without loss of effectiveness. They include exploring new search heuristics and scoring functions to guide the search, exploring conditions in small chunks, sorting the features so that the most frequent in the past are explored first, and so on.

Acknowledgments. Our work was funded by the Spanish and the Andalusian R&D-&I programmes by means of grants TIN2007-64119, P07-TIC-2602, P08-TIC-4100, TIN2008-04718-E, TIN2010-21744, TIN2010-09809-E, TIN2010-10811-E, TIN2010-099 88-E, TIN2011-15497-E, and TIN2013-40848-R, which got funds from the European FEDER programme.

References

1. Álvarez, M., Pan, A., Raposo, J., Bellas, F., Cacheda, F.: Finding and extracting data records from web pages. Signal Process. Syst. **59**(1), 123–137 (2010)
2. Arasu, A., Garcia-Molina, H.: Extracting structured data from web pages. In: SIGMOD Conference, pp. 337–348 (2003)
3. Ashraf, F., Özyer, T., Alhajj, R.: Employing clustering techniques for automatic information extraction from HTML documents. IEEE Trans. Syst. Man Cybern. Part C **38**(5), 660–673 (2008)
4. Barbosa, J.P.D.: Adaptive record extraction from web pages. In: WWW, pp. 1335–1336 (2007)
5. Bădică, C., Bădică, A., Popescu, E., Abraham, A.: L-wrappers: concepts, properties and construction. Soft Comput. **11**(8), 753–772 (2007)
6. Califf, M.E., Mooney, R.J.: Bottom-up relational learning of pattern matching rules for information extraction. J. Mach. Learn. Res. **4**, 177–210 (2003)
7. Chang, C.H., Kayed, M., Girgis, M.R., Shaalan, K.F.: A survey of web information extraction systems. IEEE Trans. Knowl. Data Eng. **18**(10), 1411–1428 (2006)
8. Chang, C.H., Kuo, S.C.: OLERA: semisupervised web-data extraction with visual support. IEEE Intel. Syst. **19**(6), 56–64 (2004)
9. Cohen, W.W., Hurst, M., Jensen, L.S.: A flexible learning system for wrapping tables and lists in HTML documents. In: WWW, pp. 232–241 (2002)
10. Crescenzi, V., Mecca, G.: Automatic information extraction from large websites. J. ACM **51**(5), 731–779 (2004)
11. Crescenzi, V., Merialdo, P.: Wrapper inference for ambiguous web pages. Appl. Artif. Intel. **22**(1–2), 21–52 (2008)
12. Fernández-Villamor, J.I., Iglesias, C.A., Garijo, M.: First-order logic rule induction for information extraction in web resources. Int. J. Artif. Intel. Tools **21**(6), 20 (2012)
13. Freitag, D.: Machine learning for information extraction in informal domains. Mach. Learn. **39**(2/3), 169–202 (2000)
14. Gregg, D.G., Walczak, S.: Exploiting the information web. IEEE Trans. Syst. Man Cybern. Part C **37**(1), 109–125 (2007)
15. Gulhane, P., Madaan, A., Mehta, R.R., Ramamirtham, J., Rastogi, R., Satpal, S., Sengamedu, S.H., Tengli, A., Tiwari, C.: Web-scale information extraction with vertex. In: ICDE, pp. 1209–1220 (2011)
16. Hogue, A.W., Karger, D.R.: Thresher: automating the unwrapping of semantic content from the world wide web. In: WWW, pp. 86–95 (2005)
17. Hsu, C.N., Dung, M.T.: Generating finite-state transducers for semi-structured data extraction from the Web. Inf. Syst. **23**(8), 521–538 (1998)
18. Irmak, U., Suel, T.: Interactive wrapper generation with minimal user effort. In: WWW, pp. 553–563 (2006)
19. Kayed, M., Chang, C.H.: Fivatech: page-level web data extraction from template pages. IEEE Trans. Knowl. Data Eng. **22**(2), 249–263 (2010)

20. Kosala, R., Blockeel, H., Bruynooghe, M., den Bussche, J.V.: Information extraction from structured documents using k-testable tree automaton inference. Data Knowl. Eng. **58**(2), 129–158 (2006)
21. Kushmerick, N., Weld, D.S., Doorenbos, R.B.: Wrapper induction for information extraction. In: IJCAI, vol. 1, pp. 729–737 (1997)
22. Liu, B., Zhai, Y.: NET – a system for extracting web data from flat and nested data records. In: Ngu, A.H.H., Kitsuregawa, M., Neuhold, E.J., Chung, J.-Y., Sheng, Q.Z. (eds.) WISE 2005. LNCS, vol. 3806, pp. 487–495. Springer, Heidelberg (2005)
23. Liu, W., Meng, X., Meng, W.: Vide: a vision-based approach for deep web data extraction. IEEE Trans. Knowl. Data Eng. **22**(3), 447–460 (2010)
24. Meng, W., Yu, C.T.: Advanced Metasearch Engine Technology. Morgan & Claypool Publishers, USA (2010)
25. Muslea, I., Minton, S., Knoblock, C.A.: Hierarchical wrapper induction for semi-structured information sources. Auton. Agents Multi-Agent Syst. **4**(1/2), 93–114 (2001)
26. Raposo, J., Pan, A., Álvarez, M., Hidalgo, J., Viña, Á.: The wargo system: semi-automatic wrapper generation in presence of complex data access modes. In: DEXA Workshops, pp. 313–320 (2002)
27. Simon, K., Lausen, G.: ViPER: augmenting automatic information extraction with visual perceptions. In: CIKM, pp. 381–388 (2005)
28. Sleiman, H.A., Corchuelo, R.: An unsupervised technique to extract information from semi-structured web pages. In: Wang, X.S., Cruz, I., Delis, A., Huang, G. (eds.) WISE 2012. LNCS, vol. 7651, pp. 631–637. Springer, Heidelberg (2012)
29. Sleiman, H.A., Corchuelo, R.: A survey on region extractors from web documents. IEEE Trans. Knowl. Data Eng. **25**(9), 1960–1981 (2013)
30. Sleiman, H.A., Corchuelo, R.: TEX: an efficient and effective unsupervised web information extractor. Knowl.-Based Syst. **39**, 109–123 (2013)
31. Sleiman, H.A., Corchuelo, R.: A class of neural-network-based transducers for web information extraction. Neurocomputing **135**, 61–68 (2014)
32. Sleiman, H.A., Corchuelo, R.: Trinity: on using trinary trees for unsupervised web data extraction. IEEE Trans. Knowl. Data Eng. **26**(6), 1544–1556 (2014)
33. Su, W., Wang, J., Lochovsky, F.H.: ODE: ontology-assisted data extraction. ACM Trans. Database Syst. **34**(2) (2009)
34. Tao, C., Embley, D.W.: Automatic hidden-web table interpretation, conceptualization, and semantic annotation. Data Knowl. Eng. **68**(7), 683–703 (2009)
35. Turmo, J., Ageno, A., Català, N.: Adaptive information extraction. ACM Comput. Surv. **38**(2) (2006)
36. Wang, J., Lochovsky, F.H.: Data extraction and label assignment for web databases. In: WWW, pp. 187–196 (2003)
37. Zhai, Y., Liu, B.: Structured data extraction from the web based on partial tree alignment. IEEE Trans. Knowl. Data Eng. **18**(12), 1614–1628 (2006)
38. Zhu, J., Nie, Z., Wen, J.R., Zhang, B., Ma, W.Y.: Simultaneous record detection and attribute labeling in web data extraction. In: KDD, pp. 494–503 (2006)