A Novel Approach to Web Information Extraction

Antonia M. Reina Quintero(✉), Patricia Jiménez, and Rafael Corchuelo

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

Abstract. Business Intelligence requires the acquisition and aggregation of key pieces of knowledge from multiple sources in order to provide valuable information to customers. The Web is the largest source of information nowadays. Unfortunately, the information it provides is available in semi-structured human-friendly formats, which makes it difficult to be processed by automated business processes. Classical propositional and ILP machine-learning techniques have been applied for this purpose. However, the former have not enough expressive power, whereas the latter are more expressive but intractable with large datasets. Propositionalisation was devised as a means to provide propositional techniques with more expressive power, enabling them to exploit structural information in a propositional way that allows them to be efficient. In this paper, we present a proposal to extract information from semi-structured web documents that uses this approach. It leverages a classical propositional machine learning technique and enhances it with the ability to learn from an unbounded context, which helps increase its precision and recall. Our experiments prove that our proposal outperforms other state-of-art techniques in the literature.

1 Introduction

Business Intelligence can be defined as the process of finding, gathering, aggregating, and analysing information for decision making [15]. The World Wide Web is an increasingly important data source for business decision making; however, extracting information from the Web remains one of the challenging issues related to Web business intelligence applications [6]. Web business intelligence is an emerging type of decision support software that "leverages the unprecedented content on the Web to extract actionable knowledge in an organizational setting" according to [20].

Web business intelligence applications should have access to structured data that could be easily extracted and incorporated into decision-making applications. However, the Web provides textual information in semi-structured formats, but those formats are intended for human consumption, which makes them difficult to be processed automatically for business purposes. Web information extractors are tools designed to extract the information of interest from such documents in a structured format.

W. Abramowicz (Ed.): BIS 2015, LNBIP 208, pp. 152–161, 2015.
DOI: 10.1007/978-3-319-19027-3_13

We focus on supervised rule-based information extractors, which are general-purpose algorithms that are configured by means of extraction rules that are learnt from labelled examples that must be provided by a user. Such rules are actually classifiers, since they get a piece of text or a DOM node as input and predict a user-defined class for it.

Unfortunately, many existing proposals build on specific-purpose machine-learning techniques that were specifically tailored to the problem of extracting web information. This makes it difficult to adapt them as the Web evolves because the features of the documents on which they rely and the techniques used to analyse them are built into the proposals.

Machine-learning techniques that come from the field of concept learning have proven to be the only approaches that can scale along a number of distinct sources and can infer knowledge in a general way. There exists two approaches: relational and propositional learning. The former attempts to learn first-order rules that generalise a set of examples by using rich representations that allow to exploit both attributive features of the examples and their relationships, whereas the latter only use attributive features. Therefore, relational systems are preferable since they allow to learn more expressive rules than their counterparts [9]. Unfortunately, a major drawback of relational systems is the expensive learning process, which becomes intractable with large datasets [1].

Propositionalisation [11] is a different way to conduct relational learning efficiently. It maps a relational representation onto features in propositional form, that is, in an attribute-value representation that is amenable for conventional data mining systems. Although propositionalisation has some inherent limitations, such as the inability to learn recurrent definitions, there are several reasons for using this process, the most important ones are: it deals with the combinatorial explosion of the number of potential features; it can leverage the existing knowledge regarding propositionalisation; and, finally, our results prove that our approach is very competitive in terms of effectiveness and efficiency since it outperforms the results obtained by classical and new proposals in the literature for web information extraction.

To the best of our knowledge, no propositionalisation system has been developed in the field of web information extraction. In this paper, we present the first such proposal in the literature. We have performed an extensive experimentation that proves that it outperforms other state-of-the-art proposals, which proves that it is very promising.

The intuitive idea behind our approach, named Yebra, is the following: it relies on a training set that consists of a set of nodes to extract and a subset of nodes that should not be extracted. Then, it learns the best possible rules by using only the attributive features of the nodes in the training set. If the rules thus learnt are perfect, the process finishes. Otherwise, Yebra tries to find better rules by combining attributive features of the nodes in the training set with attributive features of their neighbour nodes. Neighbour nodes are reached by applying relational features such as next sibling or parent, amongst others. At each step and whilst no perfect rules are found, the system continues combining attributive features from the previous step with attributive features of increasingly distant neighbour nodes.

Algorithm 1. YEBRA's main procedure.

```
1: procedure YEBRA(dataSet, relations)
2:     trainingSet ← createTrainingSet(dataSet)
3:     testSet ← dataSet
4:     resultRule ← learnRule(trainingSet, testSet)
5:     resultBindings ← {(node, null, null)}
6:     expansion ← resultRule
7:     while ¬isPerfect(resultRule, testSet) and expansion ≠ null do
8:         (expansion, resultBindings, trainingSet, testSet) ←
                        findExpansion(resultRule, resultBindings, trainingSet,
                        testSet, dataSet, relations)
9:         if expansion ≠ null then
10:            resultRule ← expansion
11:        end if
12:    end while
13:    return (resultRule, resultBindings)
14: end procedure
```

The rest of the paper is organised as follows: Sect. 2 explains our proposal; then the experimental results are detailed in Sect. 3; Sect. 4 presents the related work; Sect. 5 concludes the paper.

2 Description of Our Proposal

In this section, we first describe our training and test sets, then report on Yebra's main algorithm, and then on the algorithms to learn a rule and to expand it.

2.1 The Training Set and the Test Set

The training set is a bag of vectors. Each vector refers to a node in the DOM tree, for which it holds its set of attributive features (A_{ij}) and its class (C_i). An initial training set is created by the *createTrainingSet* function (Line 2), which works as follows: first, it adds the positive examples to the training set. Positive examples are those vectors whose class is not *null*. Then, it adds some negatives examples (not all) to the training set. The negative examples are the ones whose class is *null*. The negative examples added to the training set are computed amongst the negative examples that are the nearest neighbours to the positive examples that belong to the dataset. That is, for every positive example, Yebra finds its k nearest neighbours and adds them to the training set. Note that this process helps Yebra to better discover the frontier amongst positive and negative examples. Experimentally, we have found out that setting k to three is a good trade off between efficiency and effectiveness.

2.2 The Main Procedure

Algorithm 1 presents the main procedure in Yebra. The first step is to find the best rule using attributive features only (Lines 2–4). Then, it iterates until the rule being learnt is perfect or there are not any further expansions (Line 7). An expansion is a new rule learnt by adding extra information to the training set.

The extra information is added as new components to the vectors of the training set (and also the test set). The new components are the attributive features of those nodes that are reachable by applying a relational feature or a combination of relational features. Thus, the existence of an expansion means that Yebra has found a rule that improves on the previous one. But it has to check if this new rule can also be improved (that is, if a new, better classifier can be learnt by adding more features of the neighbours to the training set). If the rule cannot be improved, then there are no expansions, which means that Yebra has found the best possible rule. Finally, note that in each iteration *resultRule* contains the best rule found so far. If it is not perfect, but can be improved, Yebra continues with the next iteration. If it is not perfect and cannot be improved, Yebra stops and returns it.

2.3 Learning a Rule

Once the initial training and test sets are created, Yebra has to learn the best possible rule using procedure *learnRule* (Algorithm 2). It works on a training set and a test set and returns a rule. The intuitive idea behind this procedure is the following: first, it learns a rule, if the rule is perfect, then the procedure stops and returns that rule; otherwise, the training set is modified by adding the misclassified examples, and the process is repeated until the rule is perfect or the rule does not improve on the previous one. That is, the training set is enriched iteratively with misclassified examples, which helps learn better rules.

The procedure starts by balancing the training set (Line 2). This step is necessary because function *learnClassifier* (Line 3) uses a PART learner [21], which does not have good results with unbalanced datasets. The *balance* function computes the number of elements of the majority class, and it then replicates as many examples of each class as needed to obtain a training set that has approximately the same number of examples of every class.

After that, it learns a PART classifier from the balanced training set (Line 3). Then, the set of misclassified examples is computed using this classifier and the test set. Note that although the classifier has been learnt from the input training set (which is a subset of the nodes in the original dataset), the misclassified examples are computed from the test set (which includes all of the nodes in the original dataset).

Then, the procedure iterates until the rule is perfect (the misclassified set is empty) or the new rule does not improve on the old rule. In each iteration:

1. A new training set is computed. It is the union of the vectors included in the previous training set and the vectors related to the nodes that have been misclassified.
2. A new classifier is learnt from the new training set.
3. A new set of misclassified examples is computed for the new classifier.

If the new classifier is perfect, the procedure ends and returns the new classifier. If the new classifier is not perfect and does not improve on the old classifier, the

Algorithm 2. Procedure LEARNRULE

```
1: procedure LEARNRULE(trainingSet, testSet)
2:     oldTrainingSet ← balance(trainingSet)
3:     oldRule ← learnClassifier(oldTrainingSet)
4:     oldMisclassified ← findMisclassified(oldRule, testSet)
5:     isPerfect ← oldMisclassified = ∅
6:     improves ← true
7:     while ¬isPerfect and improves do
8:         newTrainingSet ← balance(oldTrainingSet ∪ oldMisclassified)
9:         newRule ← learnClassifier(newTrainingSet)
10:        newMisclassified ← findMisclassified(newRule, testSet)
11:        isPerfect ← oldMisclassified = ∅
12:        improves ←| newMisclassified |<| oldMisclassified |
13:        if isPerfect or improves then
14:            oldTrainingSet ← newTrainingSet
15:            oldRule ← newRule
16:            oldMisclassified ← newMisclassified
17:        end if
18:    end while
19:    result ← oldRule
20:    return result
21: end procedure
```

procedure ends and returns the old classifier. Finally, if the new classifier is not perfect but improves on the old one, the process is repeated to see if the new classifier can be improved. (Rule R_1 improves on rule R_2 if R_1 misclassifies less examples than R_2.)

2.4 Expanding a Rule

Once Yebra has learnt the best rule using only attributive features, it tries to improve it by adding extra information to the training set. The extra information is obtained by navigating trough the DOM tree by means of relational features, in such a way that if N_i is a vector belonging to the training set with components (A_{i1}, A_{i2},...,A_{in},C_i), and N_j is a vector belonging to the dataset with components (A_{j1}, A_{j2},...,A_{jn},C_j), and R_p is a relational feature such that $N_j = R_p(N_i)$, then an expansion of N_i using R_p is the vector (A_{i1}, A_{i2},...,A_{in},A_{j1}, A_{j2},...,A_{jn},C_i).

To deal with expansions Yebra relies on so-called bindings. A binding is a triplet $(target, relation, source)$ in which *target* and *source* are variable names, and *relation* refers to a relational feature. We can think of a binding as an expression similar to $target \leftarrow relation(source)$, that is, a binding maps a source node onto a target node using a relational feature.

The procedure responsible for expanding the rule, and as a consequence, the training and test sets, is *findExpansion* (Algorithm 3).

findExpansion works on a rule, a set of bindings, the training set, the test set, the original dataset, and a set of relational features. If *findExpansion*

Algorithm 3. Procedure FINDEXPANSION

```
1: procedure FINDEXPANSION(rule, bindings, trainingSet, testSet, dataSet, relations)
2:     resultRule ← null
3:     resultBindings ← null
4:     resultTrainingSet ← null
5:     resultTestSet ← null
6:     for all binding b in bindings while ¬isPerfect(resultRule, testSet) do
7:         for all relation r in relations while ¬isPerfect(resultRule, testSet) do
8:             newBinding ← (freshVariable(), r, target(b))
9:             newTrainingSet ← expand(trainingSet, newBinding, dataset, relations)
10:            newTestSet ← expand(testSet, newBinding, dataset, relations)
11:            newRule ← learnRule(newTrainingSet, newTestSet)
12:            if gain(rule, testSet, newRule, newTestSet) >
                          gain(rule, testSet, resultRule, resultTestSet) then
13:                resultRule ← newRule
14:                resultBindings ← bindings ∪ {newBinding}
15:                resultTrainingSet ← newTrainingSet
16:                resultTestSet ← newTestSet
17:            end if
18:        end for
19:    end for
20:    return (resultRule, resultBindings, resultTrainingSet, resultTestSet)
21: end procedure
```

cannot find an expansion that improves on the input rule, it returns a *null* expansion; otherwise, it expands the training and test sets, the bindings, and returns the new, improved, rule.

Yebra starts with the initial binding $(node, null, null)$ (Algorithm 1, Line 5), that is, the algorithm starts with a variable named *node*, and no relation involved. So the first time $findExpasion$ is executed, the set of bindings has only one element, the initial binding.

First, $findExpansion$ initialises with *null* values the variables $resultRule$, $resultBindings$, $resultTrainingSet$, and $resultTestSet$. Note that if no expansions can be found, $resultRule$ remains with a *null* value. Then it iterates on the Cartesian product of bindings and relations. For each pair:

1. It creates a new binding. The new binding is the result of applying the relational feature to that binding.
2. It expands the training set. This implies the expansion of every vector in the training set, by applying the relation specified by the new binding.
3. It expands the test set. This implies the expansion of every vector in the test set, by applying the relation defined in the new binding.
4. It learns a new rule with the expanded training and test sets. Note that we are learning the best possible rule for the expanded training set.
5. It checks whether the new rule improves on the previous one. In that case, it stores the new rule, the expanded training and test set, and the new binding.

To evaluate if the rule learnt by means of an expansion is better than the rule obtained using another different expansion, Yebra relies on the information gain function [14].

Thus, when the *findExpansion* procedure is called, the best rule obtained by expanding the training set is computed (if there is one). After that, Yebra calls *findExpansion* again to check further expansions can improve on the current rule.

3 Experimental Analysis

To prove that our proposal is worth from a practical point of view, we have performed a series of experiments in which we have collected the usual effectiveness measures, namely: precision (P), recall (R), and the $F1$ measure ($F1$). We have developed a Java 1.7 implementation of Yebra and we have run it on a four-threaded Intel Core i7 computer that ran at 2.93 GHz, had 4 GiB of RAM, Windows 7 Pro 64-bit, Oracle's Java Development Kit 1.7.0_02, and Weka 3.6.8. We used a collection of 23 datasets that provides 850 web documents regarding books, movies, conferences, cars, doctors, sports, restaurants, and so on.

The empirical comparison was performed with two classical proposals (SoftMealy [7] and WIEN [12]) and three recent proposals (RoadRunner [4], FiVaTech [10], and Trinity [18]). Regarding Yebra, we selected six documents from each site to learn an extraction rule, and then validated the result using the remaining documents; in average, we used $22.78 \pm 9.87\,\%$ of the datasets for training purposes and the rest for validation purposes.

Table 1 summarises our results. A dash in a cell means that the corresponding proposal was not able to learn a rule for a given dataset, be it because it failed to find it, because it ran out of memory, or a bug that raised an exception. It is not surprising at all that the recent proposals outperform the classical ones regarding every effectiveness measure. Amongst the recent proposals, Trinity seems to be the one that achieves the best effectiveness. Note, however, that Yebra outperforms them all since it is able to induce rules that are more precise and have better recall in general; furthermore, the standard deviation of precision and recall is smaller, which means that our learning approach is more generally stable than the others, that is, it does not generally produce rules whose effectiveness largely deviates from the average. These results prove that Yebra is quite an effective approach to web information extraction.

4 Related Work

The majority of supervised techniques in the literature build on ad-hoc machine learning algorithms that were specifically tailored to the problem of learning extraction rules [3,16]. Most of them try to learn token or XPath patterns that are based on token lexemes, their lexical classes, or HTML tags and their attributes: [12] presented a pioneering technique to learn two patterns of tokens that characterise the left and the right context of the information to extract. Hsu and Dung [7] devised an approach that first models the structure of the information in a set of web documents using finite automata and then learns transition conditions. Soderland [19] presented a proposal that starts with an overly-general

Table 1. Experimental results

	SoftMealy			WIEN			RoadRunner			FivaTech			Trinity			Yebra		
Summary	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
Mean	0.84	0.57	0.62	0.81	0.67	0.71	0.42	0.42	0.42	0.81	0.83	0.81	0.95	0.91	0.92	0.97	0.94	0.95
Standard deviation	0.16	0.32	0.30	0.16	0.26	0.22	0.46	0.47	0.45	0.18	0.18	0.17	0.08	0.16	0.12	0.04	0.09	0.06
Dataset	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
Awesome Books	1.00	0.39	0.56	0.77	0.26	0.39	1.00	1.00	1.00	0.85	1.00	0.92	1.00	0.87	0.93	0.97	0.90	0.93
Albania Movies	0.85	0.40	0.54	0.87	0.24	0.38	0.81	1.00	0.89	0.82	0.81	0.81	0.95	0.98	0.96	0.95	0.95	0.95
All Conferences	0.99	0.25	0.40	0.80	0.40	0.53	-	-	-	0.84	0.90	0.87	0.98	0.99	0.99	0.92	0.84	0.88
Amazon (cars)	0.98	1.00	0.99	0.97	1.00	0.98	0.27	0.33	0.30	0.60	0.67	0.63	0.93	0.73	0.82	0.97	0.90	0.93
Amazon (popartist)	0.94	0.72	0.82	0.92	0.58	0.71	0.99	0.99	0.99	1.00	1.00	1.00	1.00	0.98	0.99	1.00	1.00	1.00
Ame. Medical Assoc.	0.79	0.39	0.53	0.60	0.60	0.60	-	-	-	-	-	-	0.98	1.00	0.99	0.99	0.99	0.99
ATP World Tour	0.94	0.94	0.94	0.71	0.71	0.71	0.00	0.00	0.00	0.99	0.88	0.93	0.97	0.99	0.98	0.98	0.99	0.99
Better World Books	0.98	0.99	0.98	0.43	0.35	0.39	0.00	0.00	0.00	0.99	0.96	0.97	0.99	1.00	0.99	1.00	1.00	1.00
Car Max	0.89	0.89	0.89	0.88	0.88	0.88	0.00	0.00	0.00	0.45	0.89	0.60	1.00	1.00	1.00	1.00	0.99	0.99
Car Zone	0.92	0.02	0.05	0.82	0.83	0.83	0.00	0.00	0.00	0.92	1.00	0.96	0.98	1.00	0.99	0.98	0.99	0.99
Disney Movies	0.97	0.97	0.97	0.72	0.72	0.72	0.00	0.00	0.00	0.71	0.67	0.69	1.00	1.00	1.00	0.94	0.91	0.93
Homes	0.80	0.79	0.79	0.92	0.92	0.92	0.00	0.00	0.00	-	-	-	0.99	0.99	0.99	1.00	1.00	1.00
IAF	0.37	0.43	0.40	1.00	1.00	1.00	0.00	0.00	0.00	0.53	0.69	0.60	0.84	0.38	0.52	0.92	0.70	0.79
Insight into Diversity	0.57	0.47	0.52	1.00	1.00	1.00	0.70	0.70	0.70	1.00	0.74	0.85	0.83	0.83	0.83	0.87	0.83	0.85
Job of Mine	0.75	0.03	0.06	0.50	0.50	0.50	0.86	1.00	0.93	-	-	-	0.86	1.00	0.93	0.97	0.97	0.97
LA Weekly	0.87	0.49	0.63	0.90	0.80	0.85	0.00	0.00	0.00	0.83	0.57	0.68	0.97	0.92	0.94	0.98	0.98	0.98
Major League	0.99	0.46	0.63	0.65	0.33	0.44	0.00	0.00	0.00	0.99	1.00	0.99	0.98	0.55	0.70	0.99	0.70	0.82
Mbendi	0.60	0.60	0.60	0.80	0.40	0.53	0.90	1.00	0.95	0.90	1.00	0.95	1.00	1.00	1.00	1.00	1.00	1.00
Okra	0.83	0.82	0.82	0.60	0.67	0.63	0.96	0.56	0.71	0.49	0.34	0.40	1.00	0.82	0.90	1.00	0.95	0.97
Remax	0.84	0.84	0.84	1.00	1.00	1.00	-	-	-	-	-	-	0.70	0.98	0.82	0.85	0.97	0.91
RPM Find	0.72	0.03	0.06	0.99	0.99	0.99	0.98	0.99	0.98	-	-	-	0.95	0.97	0.96	1.00	1.00	1.00
Steady Health	0.75	0.25	0.38	0.75	0.75	0.75	0.00	0.00	0.00	0.83	0.83	0.83	1.00	1.00	1.00	1.00	1.00	1.00
UEFA (players)	0.92	0.90	0.91	0.92	0.51	0.66	0.92	0.92	0.92	0.91	0.94	0.92	1.00	0.90	0.95	1.00	1.00	1.00

rule and then specialises it with patterns that match token sequences. Muslea et al. [13] presented a proposal that builds on a hierarchical schema that models the information in a web document; for every positive example, it attempts to learn an automata that can recognise the sequence of tokens from the starting of the parent example until its initial token, and an automata that recognises the sequence of tokens from the final token until the final token of the parent example. Sleiman and Corchuelo [17] presented the most recent proposal in this field; they use finite automata to represent the structure of the information to be extracted and then use neural networks to learn transition conditions.

None of the previous techniques have explored the idea of tackling information extraction using features of the tokens or the DOM trees themselves, e.g., their length, their colour, their depth, the ratio of letters, and the like; neither have they explored using features of the nodes in the neighbourhood. The only proposals that have explored this idea rely on first-order learning procedures, namely: SRV [5], which works on the textual view of the documents; it starts with the most general rule and then specialises it by adding conditions so that the rule matches as many positive examples as possible and reduces the number of negative ones matched. Irmak and Suel [8] presented a proposal that works on the DOM-tree view of the documents and their rules are sets of conditions that work on XPaths; their proposal creates several sets of extraction rules that generalise the user-annotated examples in different ways; next, the user has to

select the set of records that best suits his or her interests, and the learning process is executed again to correct mistakes. Bădică et al. [2] presented a proposal that also works on the DOM-tree view of the input documents; their rules basically attempt to classify positive examples by means of their tags and the tags of their neighbouring nodes; their algorithm relies on the FOIL system [14] to learn a set of Horn clauses from a logic representation of the DOM-tree nodes and their features.

Our contribution to the state of the art is twofold: on the one hand, we use a propositional approach that relies on a extensive catalogue of features; on the other hand, our propositional approach has the ability to exploit relational features that explore an unbounded context in the neighbourhood of the DOM tree nodes to extract. None of these approaches has been explored so far in the literature regarding web information extraction, but our results prove that they are very promising.

5 Conclusions

In this paper, we have presented a new approach to information extraction; contrarily to the majority of proposals in the literature, which provide ad-hoc algorithms, our proposal maps the problem of information extraction onto the problem of learning from a set of vectors that represent the features computed on a DOM tree plus a number of relations amongst them. Our empirical results prove that our approach is very promising since it achieves better effectiveness than other state-of-the-art proposals.

Acknowledgements. 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, TIN2010-09988-E, TIN2011-15497-E, and TIN2013-40848-R, which got funds from the European FEDER programme.

References

1. Blockeel, H., Raedt, L.D., Jacobs, N., Demoen, B.: Scaling up inductive logic programming by learning from interpretations. Data Min. Knowl. Discov. **3**(1), 59–93 (1999)
2. Bădică, C., Bădică, A., Popescu, E., Abraham, A.: L-Wrappers: concepts, properties and construction. Soft Comput. **11**(8), 753–772 (2007)
3. 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)
4. Crescenzi, V., Merialdo, P.: Wrapper inference for ambiguous web pages. Appl. Artif. Intell. **22**(1&2), 21–52 (2008)
5. Freitag, D.: Machine learning for information extraction in informal domains. Mach. Learn. **39**(2/3), 169–202 (2000)
6. Gregg, D.G., Walczak, S.: Exploiting the information web. IEEE Trans. Syst. Man Cybern. Part C **37**(1), 109–125 (2007)

7. 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)
8. Irmak, U., Suel, T.: Interactive wrapper generation with minimal user effort. In: WWW, pp. 553–563 (2006)
9. Kavurucu, Y., Senkul, P., Toroslu, I.H.: A comparative study on ILP-based concept discovery systems. Expert Syst. Appl. **38**(9), 11598–11607 (2011)
10. Kayed, M., Chang, C.H.: FiVaTech: page-level web data extraction from template pages. IEEE Trans. Knowl. Data Eng. **22**(2), 249–263 (2010)
11. Kramer, S., Lavrač, N., Flach, P.: Propositionalization approaches to relational data mining. In: Džeroski, S., Lavrač, N. (eds.) Relational Data Mining, pp. 262–291. Springer, Heidelberg (2001)
12. Kushmerick, N., Weld, D.S., Doorenbos, R.B.: Wrapper induction for information extraction. In: IJCAI (1), pp. 729–737 (1997)
13. Muslea, I., Minton, S., Knoblock, C.A.: Hierarchical wrapper induction for semi-structured information sources. Auton. Agent. Multi-Agent Syst. **4**(1/2), 93–114 (2001)
14. Quinlan, J.R., Cameron-Jones, R.M.: Induction of logic programs: FOIL and related systems. New Gener. Comput. **13**(3&4), 287–312 (1995)
15. Saggion, H., Funk, A., Maynard, D., Bontcheva, K.: Ontology-based information extraction for business intelligence. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 843–856. Springer, Heidelberg (2007)
16. Sleiman, H.A., Corchuelo, R.: A survey on region extractors from web documents. IEEE Trans. Knowl. Data Eng. **25**(9), 1960–1981 (2013)
17. Sleiman, H.A., Corchuelo, R.: A class of neural-network-based transducers for web information extraction. Neurocomputing **135**, 61–68 (2014)
18. 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)
19. Soderland, S.: Learning information extraction rules for semi-structured and free text. Mach. Learn. **34**(1–3), 233–272 (1999)
20. Srivastava, J., Cooley, R.: Web business intelligence: mining the web for actionable knowledge. INFORMS J. Comput. **15**(2), 191–207 (2003)
21. Witten, I.H., Frank, E.: Weka machine learning algorithms in Java. In: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations, pp. 265–320. Morgan Kauffman Publishers (2000)