On Mining DOM Trees to build Information Extractors

Gretel Fernández, Hassan A. Sleiman, Rafael Corchuelo
University of Seville
Departamento LSI, ETSI Informática
Avd. Reina Mercedes S/N, Sevilla, Spain, 41012
Email: {gretel, hassansleiman, corchu}@us.es

Rafael Z. Frantz
UNIJUÍ University
Department of Technology
Rua do Comércio, 3000, Ijuí, 98700-000, RS, Brazil
Email: rzfrantz@unijui.edu.br

***Abstract*—The Web is the largest information repository. The information it contains is usually available in human-friendly formats. Companies are interested in using this information. The problem is that they need it in structured formats so that they can use it in automated business processes. In the literature, there are many proposals to infer information extractors. They build on machine learning techniques that attempt to infer a pattern in the HTML or XPath sources. To the best of our knowledge, no-one has ever explored using datamining techniques on DOM trees. In this paper, we report on a methodology that builds on datamining CSS features and a few other DOM features. Our results prove that this methodology is promising.**

***Index Terms*—Datamining Techniques, Information Extractor, Machine Learning**

I. Introduction

The web is an enormous and increasing source of information. It has been defined by some authors as the biggest store of knowledge of the humanity and the reason is that the information is usually available in a friendly format for the users that allows it to interact easily with the Web.

The problem is when the information is necessary to use an automated business process.

Currently, the two most accepted solutions are the use of Semantic Web technologies or the use of Wrappers. The Semantic Web is an extension of the current Web where all the resources have associated annotations that represent the information in structured way [2]. The Semantic Web is not a new Web but is a complement of the current web that will help the programmers make use of the information contained in the Web more easily. The Semantic Web is advancing, but, unfortunately, it is not widely deployed nowadays. Wrappers can be an effective short-term solution. A web wrapper aims at offering an API to abstract programmers from simulating human behaviour interacting with the Web. One of the main components of a web wrapper is the information extractor. Information extractors usually are general purpose algorithms that rely on a number of extraction rules. These rules can be hand-crafted but there are a number of proposals in the literature that report on learners to infer them.

To the best of our knowledge, all of the proposals in the literature build on ad-hoc machine learning techniques; for that, the use of standard datamining techniques to extract information from web pages is novel.

In [5] we reported on the results of a preliminary experiment in this field. We labeled DOM trees with labels to indicate which nodes contained relevant information. The pages must be homogeneous, i.e., they all must be about books, films, conferences or DVDs. The reason is that our technique builds on features of their DOM tree, so they all must be generated by the same server-side template; then we used the standard Naive Bayes, C4.5, IBk, ZeroR, SVM, and VFI datamining techniques to infer a classifier that performed quite well in practice. This motivated us to keep working to develop a complete methodology to create information extractors. In our experiment, we used the DOM trees features, chiefly CSS features. Initially we use the features as are, i.e., they all had to be nominal features, but we have learned that the results can be improved by normalising some of them. For instance, feature color may have values red, #12ab, or rgb(0, 255, 180), which makes it difficult to define the domain of this feature; in our methodology, we normalise this feature by transforming it into three independent numeric variables called R, G, and B that correspond to the red, green, and blue component of every colour. In this paper, we also compare our methodology with other state-of-the-art proposals in the literature. Our results prove that our methodology achieves a precision and recall that is comparable to other proposals, but the time required is significantly better.

The rest of the paper is organised as follows: In Section §II we review briefly related work, in Section §III we explain our methodology and then in Section §IV we evaluated our methodology and shows our results. Finally, Section §V presents the conclusions and our future work regarding this methodology.

II. Related Work

Literature contains many techniques to infer extraction rules from semi structured and free text web pages. Proposals such as S-CREAM [6], KIM [10], Armadillo [3] and MnM [18] works over free text but in our work we focus on information extraction from semi structured web pages. These techniques can be classified into two main groups: a heuristic based group and a rule based group.

Heuristic based group contains those proposals to extract information that are based on predefined heuristics which can not be modified by users. Although these heuristics can be seen as rules, the difference between this group and the rule based group is that they are totally independent of the web page over which they are applied. These heuristics are not inferred by any automatic technique neither learn extraction rules. Below we describe some of these techniques briefly:

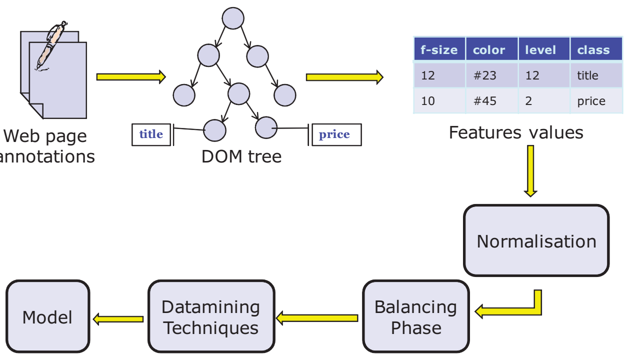
- Stavies [15] uses signal theory and clustering techniques to localise data region and extract contained records from this region.
- Alvarez et. al [1] localises data region by comparing DOM trees and then works on separating this region into records and extracted records into attributes using string alignment and other techniques.
- ViPER [16] divides input web page into visual blocks to identify data regions and then breaks them up into data records. String alignment is then used to separate attributes inside extracted data records.

The rule based group contains those information extractors that are configurable by means of rules. Beyond handcrafted information extraction rules, there are many proposals in the literature to learn them in a supervised and in an unsupervised manner. Supervised techniques require user to label a set of web pages by selecting and assigning a type for the relevant information in a set of web pages used then in the learning process. Below we describe few supervised information extraction learning proposals:

- Softmealy [8] constructs a transducer where states identify the data to extract and transitions contain the learned regular expressions to pass from one attribute to another. Transition condition are learned using a tokenisation hierarchy and an alignment technique.
- WIEN [11] is a tool that provides an implementation to six algorithms that build on the general idea of learning common prefixes and suffixes. In the rest of the paper, we focus on the NLR algorithm, which allows to extract simple fields.
- Stalker [14] uses a tree like structure called embedded catalog where each node identify and extract one type of attributes. Rules are learned by using a coverage algorithm which tries to use previous and post tokens for each annotation type, covering the maximum number of annotations and minimum number of un annotated text.
- Thresher [7] searches for sub trees inside the DOM tree similar to that one annotated by user. The annotated one is then generalised to cover all data records and the result is a sub tree which contains fixed and variant nodes.
- DEByE [12] rule learning algorithm uses these annotations to learn both: rules to extract a complete object and rules to extract attributes inside this object. Rules are inferred by searching for the largest common prefix and suffix tokens for the attributes to extract.

This group also contains unsupervised techniques which do not require the user to provide labels. Some unsupervised learning algorithms are described here:

- Fivatech [9] uses clustering technique to label HTML nodes and the alignment and pattern detection techniques to learn a grammar that defines the web page's structure.
- RoadRunner [4] compares input web pages by keeping similar string fragmentes and replacing unfixed ones by wildcards.
- DEPTA [22] uses MDR [13] to detect and separate data records distributed horizontally (in rows) or vertically (in columns) and then applies a partial tree alignment algorithm to separate attributes inside detected records.
- DeLa [20] localises data region by using [19] and then detects repeating patterns by tokenising content and using suffix tree.



f-size	color	level	class
12	#23	12	title
10	#45	2	price

Figure 1. Datamining CSS features to learn a classification model

III. Methodology

Our methodology is as follows, Figure §1:

1) Gather a collection of web pages from the site we wish to extract information.
2) Labeled the DOM trees using user-defined classes to mark the nodes that contain the relevant information, or the pre-defined class NA to mark the remaining nodes.
3) Extract the CSS features of every node, plus level in the DOM tree and length of the corresponding text. We have observed that these additional features may help in some cases.
4) Normalise the CSS features according to the categories defined in table §I.
5) Use standard datamining techniques in the literature to infer classifiers, cross-validate them, and compare the results to decide which one is the most appropriate.

The normalisation process depends on the features being analysed. We have grouped them into five categories, cf. Table §I:

Category 1 This category includes all of the features whose domain is nominal and bounded. For instance, feature font-weight ranges over domain {normal, bold, 100, 200, ..., 900}; note that no intermediate values area allowed, e.g., 110 or 850. The features in this category are not normalised.

Category 2 Here we include features whose domain is numeric, but has a few pre-defined nominal values. For instance, the range of feature vertical-align includes real numbers, and the following nominal values: {baseline, sub,

Category	CSS features
Category 1	font-style, font-variant, font-weight, background-repeat, background-attachment, text-decoration, text-transform, text-align, border-top-style, border-right-style, border-bottom-style, border-left-style, float, clear, display, white-space, background-image, list-style-image, list-style-type, list-style-position
Category 2	vertical-align, background-position, word-spacing, letter-spacing, margin-top, margin-right, margin-bottom, margin-left, padding-top, padding-right, padding-bottom, padding-left, border-top-width, border-right-width, border-bottom-width, border-left-width, width, height, text-indent, line-height
Category 3	font-size
Category 4	color, background-color, border-top-color, border-right-color, border-bottom-color, border-left-color
Category 5	font-family

Table I
CSS NORMALISATION

super, top, text-top, middle, bottom, text-bottom, auto}. These features are normalised so that nominal values are transformed into extreme real values that are not likely at all. For instance, baseline is normalised as -35000, sub is normalised as -40000, and so on.

Category 3 The features in this category may have values that are actually macro-definitions. For instance, feature font-size ranges over {xx-large, x-large, large, middle, small, x-small, xx-small} which are defined as 1.6 S, 1.4 S, 1.2 S, 1.0 S, 0.8 S, 0.4 S, and 0.2 S, where S denotes the normal font size, which is browser dependent. The normalisation consists of unrolling the macro definition.

Category 4 This category includes colour-related features. In CSS a colour can be expressed as a symbolic name, as a hexadecimal number, or in RGB notation, e.g., blue, #12ab00, rgb(12, 0, 128). These features are transformed into three independent numeric variables, each of which accounts for an independent RGB component.

Category 5 The only feature in this category is font-family. The domain of this feature is an unbounded set of sets of nominal values that includes Arial, Times New Roman, {Verdana, Helvética}, {Courier New, Tahoma} and other well-known fonts or combinations of fonts. We have identified a collection of 394 popular font names, and we have transformed this feature into a collection of 394 boolean variables: these variables are assigned true if the corresponding font family is in the value of the feature, and false otherwise. To account for other uncommon fonts, we have added a variable called other-font-family.

IV. Evaluation

To evaluate our methodology, we have conducted several experiments. We use some datasets previously annotated with the relevant information and each dataset contains thirty web pages. The following techniques were used: Naive Bayes, C4.5, IBk, ZeroR, DecisionTable, SVM [21]. We compared the results to SoftMealy, and NLR. For each technique, we measured precision (P), recall (R) and time to build model (TBM). Then, we compared the results obtained from each dataset and we made a ranking.

The analysis of our experiments rely on statistical inference. The idea is to define two hypothesis, the null hypothesis, which always considers that there are not significant differences between values and the alternative hypothesis that considers that there are differences. The goal is to reject the null hypothesis and show that there are differences. Accepting or rejecting the null depends on the comparison of the p-value calculated by the tests and significance level (α). In our study we used the standard value ($\alpha = 0.05$). If p-value is less than α then the null hypothesis is rejected.

First, we used Kolmorogov-Smirnov's test with the Lilliefors correction and the Shapiro-Wilk's test to check normality. The results show that none of the measures follow a normal distribution, cf. Table §II. The implication is that we need to resort to non-parametric tests. We used the Kruskal-Wallis test to determine if P, R and TMB means can be considered equal or not. These results proved that all techniques are statistically different and therefore it is possible make a ranking, since all of the p-values were 0.00. The next step is to use Friedman's algorithm to calculate the scores and Bergmann's algorithm to rank them. The results are shown in Table §III. We can conclude that datamining techniques we have analysed are better than the SoftMealy and the NLR techniques, which are well-known in the literature. With regard to precision, the datamining techniques are as good as the other techniques, but regarding recall and time to build models, the datamining techniques are better.

V. Conclusions

In this paper, we have presented a methodology to extract information from web pages that relies on datamining CSS features of DOM trees. To the best of our knowledge, this is the first methodology that explores this field, and the results are promising since precision and recall are comparable but the time to build models improves significantly.

In future, we plan on a) extending our methodology to extract data records, instead of just data items; exploring other techniques in the literature to balance the NA class [17]; c) exploring if using CSS features of parent or sibling nodes can help improve precision and recall, without a significant impact on the time to build models.

Technique	Kolmogorov-Smirnov (P-Value)			Shapiro-Wilk (P-Value)		
	Precision	Recall	Time	Precision	Recall	Time
Naive Bayes	0.20	0.90	0.20	0.52	0.24	0.09
C4.5	0.20	0.20	0.20	0.37	0.31	0.97
IBk	0.20	0.20	0.25	0.11	0.10	0.11
ZeroR	0.00	0.00	0.02	0.00	0.00	0.23
DecisionTable	0.20	0.20	0.20	0.16	0.61	0.71
SVM	0.03	0.62	0.00	0.34	0.21	0.03
NLR	0.14	0.20	0.50	0.64	0.49	0.04
SoftMealy	0.20	0.20	0.00	0.30	0.09	0.00

Table II
KOLMOGOROV-SMIRNOV'S AND SHAPIRO-WILK'S TESTS

Ranking	**Precision**	**Recall**	**Time to build model**
1	SoftMealy, C4.5, IBk, DecisionTable, SVM, NLR	C4.5, IBk, DecisionTable, SVM, Naive Bayes	IBk, Naive Bayes, C4.5, ZeroR
2	Naive Bayes	SoftMealy, ZeroR	SVM
3	ZeroR	NLR	DecisionTable, NLR, SoftMealy

Table III
BERGMANN'S RANKING

ACKNOWLEDGMENT

This paper was supported by the European Commission (FEDER), the Spanish and the Andalusian R&D&I programmes (grants TIN2007-64119, P07-TIC-2602, P08-TIC-4100, TIN2008-04718-E, TIN2010-21744, TIN2010-09809-E, TIN2010-10811-E, and TIN2010-09988-E).

REFERENCES

[1] M. Álvarez, A. Pan, J. Raposo, F. Bellas, and F. Cacheda. Extracting lists of data records from semi-structured web pages. *Data Knowl. Eng.*, 64(2):491–509, 2008. Available at http://dx.doi.org/10.1016/j.datak.2007.10.002

[2] G. Antoniou and F. van Harmelen. *A Semantic Web Primer, 2nd Edition*. The MIT Press, 2008

[3] F. Ciravegna, S. Chapman, A. Dingli, and Y. Wilks. Learning to harvest information for the semantic web. In *ESWS*, pages 312–326, 2004. Available at http://springerlink.metapress.com/openurl.asp?genre=article&amp;issn=0302-9743&amp;volume=3053&amp;spage=312

[4] V. Crescenzi, G. Mecca, and P. Merialdo. Roadrunner: Towards automatic data extraction from large web sites. In *VLDB*, pages 109–118, 2001. Available at http://www.vldb.org/conf/2001/P109.pdf

[5] G. Fernández and H. A. Sleiman. An experiment on using datamining techniques to extract information from the web. In *9th International Conference on Practical Applications of Agents and Multiagent Systems*, pages 169–176, 2011. Available at http://www.springerlink.com/content/k25166253816k002/

[6] S. Handschuh, S. Staab, and F. Ciravegna. S-CREAM - semi-automatic CREAtion of metadata. In *Knowledge Acquisition, Modeling and Management*, pages 358–372, 2002. Available at http://link.springer.de/link/service/series/0558/bibs/2473/24730358.htm

[7] A. Hogue and D. R. Karger. Thresher: Automating the unwrapping of semantic content from the World Wide Web. In *WWW*, pages 86–95, 2005. Available at http://doi.acm.org/10.1145/1060745.1060762

[8] C.-N. Hsu and M.-T. Dung. Generating finite-state transducers for semi-structured data extraction from the Web. *Inf. Syst.*, 23(8):521–538, 1998. Available at http://dx.doi.org/10.1016/S0306-4379(98)00027-1

[9] M. Kayed and C.-H. Chang. FiVaTech: Page-level web data extraction from template pages. *IEEE Trans. Knowl. Data Eng.*, 2010. Available at http://doi.ieeecomputersociety.org/10.1109/TKDE.2009.82

[10] A. Kiryakov, B. Popov, I. Terziev, D. Manov, and D. Ognyanoff. Semantic annotation, indexing, and retrieval. *J. Web Sem.*, 2(1):49–79, 2004. Available at http://dx.doi.org/10.1016/j.websem.2004.07.005

[11] N. Kushmerick. Wrapper induction: Efficiency and expressiveness. *Artif. Intell.*, 118(1-2):15–68, 2000. Available at http://dx.doi.org/10.1016/S0004-3702(99)00100-9

[12] A. H. F. Laender, B. A. Ribeiro-Neto, and A. S. da Silva. DEByE - data extraction by example. *Data Knowl. Eng.*, 40(2):121–154, 2002. Available at http://dx.doi.org/10.1016/S0169-023X(01)00047-7

[13] B. Liu, R. L. Grossman, and Y. Zhai. Mining web pages for data records. *IEEE Intelligent Systems*, 19(6):49–55, 2004. Available at http://csdl.computer.org/comp/mags/ex/2004/06/x6049abs.htm

[14] I. Muslea, S. Minton, and C. A. Knoblock. Hierarchical wrapper induction for semistructured information sources. *Autonomous Agents and Multi-Agent Systems*, 4(1/2):93–114, 2001

[15] N. Papadakis, D. Skoutas, K. Raftopoulos, and T. A. Varvarigou. Stavies: A system for information extraction from unknown web data sources through automatic web wrapper generation using clustering techniques. *IEEE Trans. Knowl. Data Eng.*, 17(12):1638–1652, 2005. Available at http://doi.ieeecomputersociety.org/10.1109/TKDE.2005.203

[16] K. Simon and G. Lausen. Viper: augmenting automatic information extraction with visual perceptions. In *CIKM*, pages 381–388, 2005. Available at http://doi.acm.org/10.1145/1099554.

1099672

[17] C. G. Vallejo, J. A. Troyano, and F. J. Ortega. InstanceRank: Bringing order to datasets. *Pattern Recogn. Lett.*, 31:133–142, January 2010. Available at http://portal.acm.org/citation.cfm?id=1663654.1663889

[18] M. Vargas-Vera, E. Motta, J. Domingue, M. Lanzoni, A. Stutt, and F. Ciravegna. MnM: Ontology driven semi-automatic and automatic support for semantic markup. In *Knowledge Acquisition, Modeling and Management*, pages 379–391, 2002. Available at http://link.springer.de/link/service/series/0558/bibs/2473/24730379.htm

[19] J. Wang and F. H. Lochovsky. Data-rich section extraction from HTML pages. In *Web Information Systems Engineering*, pages 313–322, 2002. Available at http://computer.org/proceedings/wise/1766/17660313abs.htm

[20] J. Wang and F. H. Lochovsky. Data extraction and label assignment for web databases. In *WWW*, pages 187–196, 2003. Available at http://doi.acm.org/10.1145/775152.775179

[21] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kauf-mann, edition 2, 2005. Available at http://www.cs.waikato.ac.nz/~ml/weka/book.html

[22] 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. Available at http://doi.ieeecomputersociety.org/10.1109/TKDE.2006.197