Self-supervised automated wrapper generation for weblog data extraction

George Gkotsis, Karen Stepanyan, Alexandra I. Cristea, and Mike Joy

Department of Computer Science
University of Warwick
Coventry CV4 7AL
United Kingdom
{G.Gkotsis,K.Stepanyan,A.I.Cristea,M.S.Joy}@warwick.ac.uk

Abstract. *Data extraction from the web is notoriously hard. Of the types of resources available on the web, weblogs are becoming increasingly important due to the continued growth of the blogosphere, but remain poorly explored. Past approaches to data extraction from weblogs have often involved manual intervention and suffer from low scalability. This paper proposes a fully automated information extraction methodology based on the use of web feeds and processing of HTML. The approach includes a model for generating a wrapper that exploits web feeds for deriving a set of extraction rules automatically. Instead of performing a pairwise comparison between posts, the model matches the values of the web feeds against their corresponding HTML elements retrieved from multiple weblog posts. It adopts a probabilistic approach for deriving a set of rules and automating the process of wrapper generation. An evaluation of the model is conducted on a dataset of 2,393 posts and the results (92% accuracy) show that the proposed technique enables robust extraction of weblog properties and can be applied across the blogosphere for applications such as improved information retrieval and more robust web preservation initiatives.*

Keywords: Web Information Extraction, Automatic Wrapper Induction, Weblogs

1 Introduction

The problem of web information extraction dates back to the early days of the web and is fascinating and genuinely hard. The web, and the blogosphere as a constituent part, correspond to a massive, publicly accessible unstructured data source. Although exact numbers of weblogs are not known, it is evident that the size of the blogosphere is large. In 2008 alone the Internet company Technorati reported to be tracking more than 112 million weblogs, with around 900 thousand blog posts added every 24 hours[1]. In Britain alone, 25% of Internet

[1] http://technorati.com/blogging/article/state-of-the-blogosphere-introduction/

users maintain weblogs or personal websites [5] that are read by estimated 77% of Web users. Hence, the volume of information published on weblogs justifies the attention of information retrieval, preservation and socio-historical research communities.

The scale is not the only challenge for capturing weblog resources. The heterogeneous nature of these resources, the large numbers of third party elements and advertisements, the rapid changes, the propagation of user-generated content and the diversity of inter-relations across the resources are among the common characteristics of the blogosphere. These characteristics amplify the complexity of capturing, processing and transforming these web resources into structured data. The successful extraction of weblog properties is of paramount importance for improving the quality of numerous applications, such as analytics, information retrieval and preservation.

Typically, the content of a weblog resides in a relational database. The automation supported by the blogging platform provides a common structure that can be observed across the various weblog pages. More specifically, the weblog post, which constitutes a building block of a weblog, comprises a set of properties, such as the title, author, publication date, post content and the categories (or tags) assigned. Whilst the data structure is presented in a consistent way across a certain weblog, it rarely appears identical across different weblogs, even if the blogging platform remains the same. The main reason for the above inconsistency is the fact that bloggers personalise the presentation of their weblog arbitrarily, hence the resulting weblog exhibits a unique appearance. Moreover, current techniques are not sufficient to meet the requirements of a weblog data extraction framework which is a)fully automated, b)high granularity and c)high quality.

One of the most prominent characteristics of weblogs is the existence of web feeds. Web feeds, commonly provided as RSS, are machine interpretable, structured XML documents that allow access to (part of) the content of a website, such as a weblog. In fact, this high quality, rigorous information contained in web feeds has already been exploited in several applications (e.g. [13]). The solution proposed here is influenced by the above idea of exploiting the web feeds and attempts to overcome the limitation of fixed number of provided post-entities. Intuitively, our approach is not to treat the web feeds as the only source of information, but as a means that allows the *self-supervised* training and generation of a wrapper automatically. During this self-supervised training session[2], the matching of the elements found between the web feeds and the weblog posts is used to observe and record the position of the cross-matched elements. Based on these observations, a set of rules is generated through an essentially probabilistic approach. These rules are later applied throughout each weblog post (regardless of its existence in the web feed).

[2] The term self-supervised is inspired by and used in a similar way by Yates et al. in order to describe their classifier induction [17]. Contrary to our approach, where we focus on data values, their approach concerns the extraction of relational information found in texts without using web feeds.

This research makes the following main contributions:

- We use web feeds for training and generating a wrapper. The generated wrapper is described in simple rules that are induced by following a probabilistic approach. We provide a simple algorithm that is noise-tolerant and takes into account the information collected about the location of HTML elements found during training.
- We make use of CSS Classes as an attribute that can supplement the more traditional XPath manipulation approach used to describe extraction rules.
- To the best of our knowledge, we are the first to propose a self-supervised methodology that can be applied on any weblog and features unprecedented levels of granularity, automation and accuracy. We support all of the above through evaluation.

The paper is structured as follows. Section 2 describes the proposed model and the methodology applied to extract the desired weblog properties. Section 3 evaluates the model, while Section 4 discusses the contribution of the approach and presents related work. Finally, Section 5 presents the conclusions.

2 Proposed Model

We adopt the definition of a wrapper proposed by Baumgartner et al. where "a wrapper is a program that identifies the desired data on target pages, extracts the data and transforms it into a structured format" [3]. As discussed above, our model aims to generate a fully automated wrapper for each weblog. The approach is divided into three steps as follows.

2.1 Step 1: Feed Processing and Capturing of Post Properties

The input for executing the first step of the proposed model involves the acquisition of the desired blog's feed content. Similarly to standard RSS readers, the model focuses on the entries that point to the weblog posts. For each entry, we search and store the attributes of *title*, *author*, *date published*, *summary* and *permalink* as the post properties.

2.2 Step 2: Generation of Filters

The second step includes the generation of filters. The naming convention we use for the concept of a *filter* is similar to the one introduced in [2], where it is described as the building block of patterns, which in turn describes a generalised tree path in the HTML parse tree. Thus, adding a filter to a pattern extends the set of extracted targets, whereas imposing a condition on a filter restricts the set of targets. The same concept is used by XWRAP [11] in order to describe the so-called "declarative information extraction rules". These rules are described in XPath-like expressions and point to regions of the HTML document that contain data records.

Following related work, we use the concept of a filter in order to identify and describe specific data elements of an HTML weblog post. Unlike previous work, where most of the tools deal with the absolute path only (for example through partial tree alignment [18]), our filters comprise a tuple, which extends existing approaches. Our approach overcomes irregularities appearing across absolute path values by providing additional, alternate means of describing the

HTML element (namely our tuple also includes CSS Classes and HTML IDs). By conducting an initial visual survey on weblogs, we hypothesize that especially CSS Classes may be used to provide an alternate and accurate way to induce extraction rules, a feature that remains unexploited in most (if not all) approaches until now. Our evaluation results support the above hypothesis.

In our approach, the filter is described using three basic attributes, as follows.

- Absolute Path: We use a notation similar to XPath's absolute path to refer to the position of the HTML element. The Absolute Path is described as a sequence of edges, where each edge is defined as the *name* of the element and the positional information of the element (*index*)[3]. This sequence of edges starts from the root of the document and ends with the element containing the value we are interested in. For example, the value */html[0]/body[1]* refers to the body element of an HTML document, since this is the second child (hence, *body[1]*) of the root HTML element (*html[0]*).
- CSS Classes: "CSS (Cascading Style Sheets) is a simple mechanism for adding style (e.g., fonts, colours, spacing) to web documents"[4], first introduced in 1996. It allows the separation of document content from document presentation through the definition of a set of rules.
- HTML ID: The ID attribute specifies a unique identifier for an HTML element of a document. It is commonly used in cases where CSS code needs to refer to one unique element (e.g. the title of a post) or run JavaScript.

Figure 1 shows the structure of a filter with an annotated example. When pointing at a specific element, a set of HTML ID values and CSS Classes together with a single-valued Absolute Path are used to describe and define the filter. More specifically, when an element is identified, any HTML IDs or CSS Classes applied on this element are added to the filter. Afterwards, an iterative selection of the parent element continues, adding HTML IDs and CSS Classes to the sets, as long as the value of the parent element contains nothing but the value identified. For the example illustrated in Figure 1, the value for the ID attribute is *single-date*, for the CSS Classes the value is *date* and the Absolute Path is *html[0]/body[1]/div[1]/div[1]/div[0]/div[0]/div[1]*.

2.3 Step 3: Induction of Rules and Blog Data Extraction

After the completion of Step 2, a collection of filters is generated for each property. When applied to the weblog posts from which they were extracted, they link back to the HTML element of the matched property. However, due to multiple occurrences of values during the text matching process of Step 2, there are cases where a value is found in more than one HTML element. This results in generating filters equal to the number of values found. Not all of the collected

[3] The positional information of an HTML element is crucial in a HTML document. This is one of the reasons that HTML DOM trees are viewed as labelled ordered trees in the literature (e.g., [7]).

[4] http://www.w3.org/Style/CSS/

IDs	CSS Classes	Absolute Path
single-date	date	html[0]/body[1]/div[1]/div[1]/div[0]/div[0]/div[1]

Fig. 1. The structure of a filter. An example is annotated for the case of an element containing a date value.

filters are suitable for extracting the data out of a weblog; the collection contains diverse and sometimes "blurred" information that needs further processing.

In the case of weblog data extraction, there is neither prior knowledge of the location of the elements to be identified, nor a definite, automated way to describe them. We propose a case-based reasoning mechanism that assesses the information found in filters. The aim of this mechanism is to generate rules through a *learning by example* methodology, i.e., a general *rule* is extracted through the observation of a set of *instances*. In our case, the *instances* correspond to the weblog posts that lead to the generation of the filters during the previous step. The *rules* are defined in the language used to describe the previously collected filters. Therefore, they describe how to extract the weblog properties. Our approach deals with irregularities found in web documents (and filters thereof) in an inherently probabilistic way.

During the step of the induction of rules, our aim is to account for each attribute of each filter (Absolute Path, CSS Classes and HTML IDs), in order to assess the usefuleness of each attribute value as a rule. Essentially, a rule is the result of the transposition of a filter. This transposition can result in maximum three rules. The rule is described by its *type* (one of the three different attribute types of the filters), a *value* (the value of the corresponding filter's attribute) and a *score*, which is used to measure its expected accuracy. An important consideration taken here is the fact that selecting a "best-match" filter (i.e. a tuple of *all* three filter attributes) from the list of the filters or a combination of values for *each* attribute (as a collection of "best-of" values for each attribute) may result in the elimination of the desired element (for example, the HTML ID might increment for each weblog post and is therefore unique for every instance). The approach adopted here is based on the assumption that a single attribute (either the Absolute Path, or the CSS Classes or the HTML IDs) should suffice for describing the extraction rule. In the evaluation section, we give evidence why using a single attribute meets this expectation.

To assess the rule that best describes the extraction process, a score is calculated for each rule. The score aims at keeping track of the effectiveness of the

Algorithm 1 Rule induction algorithm

Inputs:
Collection of training posts P, Collection of candidate rules R
Outputs:
Rule with the highest score
for all Rules $r \in R$ **do** ▷ Initialize all scores
$r.score \leftarrow 0$
end for
Rule $rs \leftarrow$ new Rule()
$rs.score \leftarrow 0$
for all Rules $r \in R$ **do**
for all Posts $p \in P$ **do** ▷ Check if application $r(p)$ of rule r, on post p succeeds
If $r(p)$ =value-property of p **then**
$r.score++$
end for
$r.score \leftarrow \frac{r.score}{|P|}$ ▷ Normalize score values
If $r.score > rs.score$ **then** ▷ Check if this is the best rule so far
$rs \leftarrow r$
end for
return rs

rule, when applied across different posts of the weblog. As seen in Algorithm 1, an iteration takes place for each of the candidate rules, which in turn is applied on each of the training posts. For each successful match, the score of the rule is increased by one[5]. After all posts have been checked, the value is divided by the number of training posts against which the rule was validated, in order to represent a more meaningful, normalised measurement (i.e. the higher the better: 1 means that rule is successful for all posts, 0 means that it failed for each of the posts applied). The rule having the highest score – if any – is returned.

Figure 2 presents an overview of the approach described above. As already discussed in detail, the proposed solution involves the execution of three steps. The first step includes the task of reading and storing the weblog properties found in the web feed. The second step includes training the wrapper through the cross matching of information found in the web feed and the corresponding HTML documents. This step leads to the generation of information, captured through the filters, which describes where the weblog data properties reside. The final step transforms the filters into rules and calculates the rule scores in order to select a rule for each of the desired properties.

2.4 Property Matching

The proposed method relies on the identification of an HTML element against a specific value. Text matching can be used for achieving the above identification. Generally, text matching is not a trivial task and can be classified into various

[5] A successful match between properties is a crucial issue in our approach and is discussed in detail in Section 2.4.

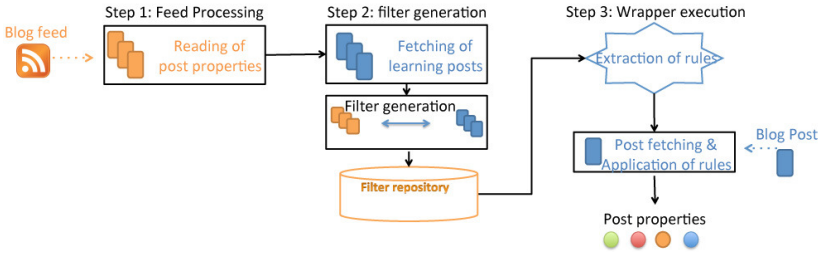

Fig. 2. Overview of the weblog data extraction approach.

string matching types. More specifically, text matching may be complete, partial, absolute or approximate. The matching of the elements is treated differently for different properties, which is another contribution of this paper (details have been omitted due to space limitations). For the title we look for absolute and complete matchings, for the content we use the Jaro-Winkler metric [15] which returns high similarity values when comparing the summary (feed) against the actual content (web page), for the date we use the Stanford NER suite for spotting and parsing the values [6], and for the author we use partial and absolute matching with some boilerplate text (i.e. "Written By" and "Posted By").

3 Evaluation

We evaluated our model against a collection of 240 weblogs (2,393 posts) for the title, author, content and publication date. For the same collection, we used the Google Blogger and WordPress APIs (in the limits of free quota) in order to get valid and full data (i.e. full post content) and followed the 10-fold validation technique [16]. As seen in Table 1, the prediction accuracy is high (mean value *92%*). For the case of the title, the accuracy is as high as *97.3%* (65 misses). For the case of the content, the accuracy is *95.9%* (99 misses). Publication date is 89.4% accurate (253 misses) and post author is 85.4% (264 misses). Table 1 summarizes the above results and presents the accuracy of Boilerpipe (77.4%) [8] (Boilerpipe is presented in detail in Section 4). Concerning the extraction of the title using Boilerpipe, the captured values are considered wrong, since the tool extracts the title of the HTML document. For the case of the main content, our model achieves *81.6%* relative error reduction. Furthermore, the overall average score for all rules is *0.89*, which presumably indicates that the induction of the selected rules is taking place at a high confidence level.

	Title	Content	Publication Date	Author
Proposed Model	97.3%(65)	95.9% (99)	89.4% (253)	85.4% (264)
Boilerpipe	0	77.4% (539)	N/A	N/A

Table 1. Results of the evaluation showing the percentage of successfully extracted properties. Number of misses are in parenthesis.

4 Discussion and Related Work

The concept of using web feeds for capturing data is not new. ArchivePress is one of the weblog archiving projects that have developed solutions for harvesting the content of weblog feeds [14]. The solution focuses solely on collecting the structured content of weblog feeds that contain posts, comments and embedded media. The solution provided by ArchivePress remains highly limited, due to the fixed number of entries and partial content (i.e. post summary) found in feeds. Another approach that attempts to exploit web feeds was developed by Oita and Sellenart [13]. This approach is based on evaluating a web page and matching it to the properties found in the corresponding web feed. The general principle of cross-matching web feeds and pages constitutes the foundation of the approach that we propose in this paper. However, because the approach by Oita and Sellenart does not devise general extraction rules, it remains inapplicable for capturing the data that are no longer available in the corresponding web feed. Additionally, the performance of their approach for extracting distinct properties such as title was reported as poor (no figures were provided in the paper).

To position our approach from a more general point of view (within the domain of earlier conducted work on web information extraction), we classify it according to the taxonomy of data extraction tools by Laender [10]. More specifically, our approach can be associated with the Wrapper Induction and Modelling-Based approaches. Similarly to the wrapper induction tools, our approach generates extraction rules. However, unlike many wrapper induction tools, our approach is fully automated and does not rely on human-annotated examples. Instead, it uses web feeds as a model that informs the process of generating extraction rules and it therefore resembles the Modelling-Based approaches. Hence, the approach presented in this paper can be positioned in relation to tools such as WIEN [9], Stalker [12], RoadRunner [4] or NoDoSE [1].

WIEN is among the first tools aimed at automating the process of information extraction from web resources. The term *wrapper induction* is, in fact, coined by the authors [9] of the tool. However, as one of the earlier attempts, the use of the tool is restricted to a specific structure of the page and the heuristics of the presented data. Furthermore, it is not designed to work with nested structure of web data. The limitation of working with hierarchical data has been addressed by the Stalker tool [12]. However, the use of the Stalker tool requires a supervised training data set that limits the degree of automation offered by the system. An attempt to automate the process of wrapper induction was made by Crescenzi et al. [4] and published along with the RoadRunner tool. The tool analyses structurally pairs of similar resources and infers an unlabelled (i.e. no property identified) schema for extracting the data. NoDoSE [1] represents a different, modelling-based category of tools that requires an existing model that defines the process of extraction. This is a semi-automatic approach due to the necessary human input for developing models. However, additional tools, such as a graphical user interface for marking resources, can be used for facilitating human input. Hence, the review of the earlier work suggests that our approach, as proposed in this paper, addresses a niche not served by the existing tools.

Among the generic solutions there are other technologies that aim at identifying the main section (e.g. article) of a web page. The open source Boilerpipe system is state-of-the-art and one of the most prominent tools for analysing the content of a web page [8]. Boilerpipe makes use of the structural features, such as HTML tags or sequences of tags forming subtrees, and employs methods that stem from quantitative linguistics. Using measures, such as average word length and average sentence length, Boilerpipe analyses the content of each web page segment and identifies the main section by selecting the candidate with the highest score. As reported by Oita and Sellenart [13], the use of Boilerpipe delivers relatively good precision (62.5%), but not as high as our approach.

Lastly, it is necessary to discuss the limitations of the proposed model and future work. First of all, a requirement for the adoption of the model is the existence and integrity of web feeds. While web feeds are prominent characteristics of weblogs, some weblogs are not configured to publish their updates through feeds. In this case, the proposed model would not be appropriate to extract any data. Additionally, a technique used to deceive anti-spam services is to report false information in web feeds. In this case, the proposed model will signal a low score on the rules – if any – generated (in fact, this limitation may be further considered for spam detection). Another limitation concerns the extraction of the *date* property. The date is currently processed for the English language only, which may pose problems when matching the date in weblogs written in different languages. An improvement would be to identify the language of the document (e.g. with Apache Tika) and style the date following the locale of the identified language. Concerning future work, the approach can be altered and deployed in a supervised manner as well. In that case, the manual labelling of HTML elements will allow running the information extraction model on websites without the requirement for web feeds. Finally, another idea worth considering is to keep feeds for labelling data and to develop more robust ways of generating XPath expressions. We intend to explore the above opportunities in the future.

5 Conclusions

In this paper, we have presented a method for fully automated weblog wrapper generation. The generated wrapper exhibits increased granularity, since it manages to identify and extract several weblog properties, such as the title, author, publication date and main content of the post. This is accomplished through the induction of rules, which are selected following a probabilistic approach based on their scoring. Devising these rules is based on the generation of filters. The filters constitute a structure that, when applied to a web document, singles out an HTML element. They are described in tuples, where each of its element-attributes describes the HTML element in different forms (Absolute Path, CSS Classes and HTML IDs). The overall approach is evaluated against a real-world collection of weblogs and the results show that the wrappers generated are robust and efficient in handling different types of weblogs.

6 Acknowledgments

This work was conducted as part of the BlogForever project funded by the European Commission Framework Programme 7 (FP7), grant agreement No.269963.

References

1. B. Adelberg. NoDoSE–a tool for semi-automatically extracting structured and semistructured data from text documents. *SIGMOD Rec.*, 27(2):283–294, 1998.
2. R. Baumgartner, S. Flesca, and G. Gottlob. Visual Web Information Extraction with Lixto. In *Proceedings of the 27th International Conference on Very Large Data Bases*, pages 119–128, San Francisco, USA, 2001. Morgan Kaufmann Publishers.
3. R. Baumgartner, W. Gatterbauer, and G. Gottlob. Web data extraction system. In *Encyclopedia of Database Systems*, pages 3465–3471. Springer, 2009.
4. V. Crescenzi, G. Mecca, and P. Merialdo. Roadrunner: Towards automatic data extraction from large web sites. In *Proceedings of the international conference on Very Large Data Bases*, pages 109–118, 2001.
5. W. Dutton and G. Blank. Next generation users: The internet in Britain. 2011.
6. J. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics, 2005.
7. P. Geibel, O. Pustylnikov, A. Mehler, H. Gust, and K. Kühnberger. Classification of documents based on the structure of their DOM trees. In *Neural Information Processing*, pages 779–788. Springer, 2008.
8. C. Kohlschütter, P. Fankhauser, and W. Nejdl. Boilerplate detection using shallow text features. In *Proceedings of the third ACM international conference on Web search and data mining*, WSDM '10, pages 441–450, New York, USA, 2010. ACM.
9. N. Kushmerick. Wrapper induction: Efficiency and expressiveness. *Artificial Intelligence*, 118(1):15–68, 2000.
10. A. Laender, B. Ribeiro-Neto, A. Da Silva, and J. Teixeira. A brief survey of web data extraction tools. *ACM Sigmod Record*, 31(2):84–93, 2002.
11. L. Liu, C. Pu, and W. Han. XWrap: An extensible wrapper construction system for internet information. In *Proceedings of the 16th International Conference on Data Engineering (ICDE 2000)*, pages 611–621, San Diego, CA, March 2000. IEEE.
12. I. Muslea, S. Minton, and C. Knoblock. Hierarchical wrapper induction for semistructured information sources. *Autonomous Agents and Multi-Agent Systems*, 4(1):93–114, 2001.
13. M. Oita and P. Senellart. Archiving data objects using Web feeds. In *Proceedings of International Web Archiving Workshop*, pages 31–41, Vienna, Austria, 2010.
14. M. Pennock and R. Davis. ArchivePress: A Really Simple Solution to Archiving Blog Content. In *Sixth International Conference on Preservation of Digital Objects (iPRES 2009)*, California Digital Library, San Francisco, USA, October 2009.
15. W. E. Winkler. String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. *Proceedings of the Section on Survey Research Methods American Statistical Association*, pages 354–359, 1990.
16. I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition.* Morgan Kaufmann, 2005.
17. A. Yates, M. Cafarella, M. Banko, O. Etzioni, M. Broadhead, and S. Soderland. Textrunner: Open information extraction on the web. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 25–26, 2007.
18. Y. Zhai and B. Liu. Web data extraction based on partial tree alignment. In *Proceedings of the 14th international conference on World Wide Web*, pages 76–85. ACM, 2005.