Control and Cybernetics

vol. **41** (2012) No. 4

Automatic wrapper generation and generalization for social media websites[1]

by

Bartosz Baziński and Michał Brzezicki

Faculty of Electronics, Telecommunications and Informatics,
Gdansk University of Technology, 80-233 Gdańsk, Poland

Abstract: The data contained within user generated content websites prove to be valuable in many applications, for example in social media monitoring or in acquisition of training sets for machine learning algorithms. Mining such data is especially difficult in case of web forums, because of hundreds of various forum engines used. We propose an algorithm capable of unsupervised extraction of posts from social websites, without the need to analyse more than one page in advance. Our method localizes potential data regions by repetition analysis within document structure and filtering potential results. Subsequently, the fields of data records are found using key characteristics and series-wide dependencies. We managed to achieve 85% precision of extraction and 79% recall after experiments on single pages taken from 258 websites. Our solution is characterized by high computing efficiency, thus enabling wide applications.

Keywords: automatic wrapper generation, information extraction, social media websites, web forums

1. Introduction

1.1. Problem description

Never in the history of mankind so much data has been recorded and stored in accessible way. As the estimated size of the World Wide Web reaches 50 billion web pages(The size of the World Wide Web, http://www.worldwidewebsize.com), it seems that every aspect of our lives, every product, person or institution has been mentioned on-line. Although the WWW is an almost infinite data source, information is provided in a very incoherent way, making it hard to gather in unified database. Only a small proportion of statements can be easily accessed through public APIs (merely large social platforms, e.g. Facebook, Twitter and

[1]Submitted: October 2012; Accepted: November 2012

Google+). The vast majority of data is scattered among numerous blogs, forums, news sites, etc. Automated extraction of data from such sites is a difficult task, because each source presents information by its own, unique structure of HTML tags.

A program capable of performing such automatic extraction is called a wrapper. A *wrapper* is defined as a procedure that translates content from specific information source into a relational model (Kushmerick, 1997). In case of extraction from social media websites, we would define input to the wrapper as a text string containing HTML structure of a web page and output as a list of statements contained within that page, with content, author and date of creation for each statement. Although creation of such wrapper for a single source is a relatively easy task and can be performed manually, problems arise when we need to obtain data from thousands of diverse websites. With almost every site using different layout template, we require a tool capable of creating wrappers, entitled a wrapper generator.

1.2. Applications of wrapper generators

Development of an efficient wrapper generator - both in terms of computing performance and extraction precision - would have practical applications in numerous cases. Wrappers for web forums would allow access to enormous amount of discussions on various topics. Those posts could be mined and used to provide direct answers to questions of users in Q & A websites and Internet search engines (Cong et al., 2008).

Another field of high application potential is social media monitoring. With the explosion of Web 2.0 platforms such as blogs and discussion forums, the consumers have gained unimaginable abilities to share their opinions with others and influence their decisions. Knowing what consumers are saying can enable companies to derive valuable marketing intelligence (Glance et al., 2005) and better align all their activities within the needs of their clients (Pang & Lee, 2008). Due to increasing fragmentation of the media, the traditional, manual methods of observation and data gathering become useless (Kim, 2006). In order to automatically oversee conversations in social media, one must implement effective extraction of content. Web forums, blogs and review sites are only accessible through web crawling, hence comes the need for wrapper generator. For precise data analysis, the author, date, statement and context of a post is needed. Statements extracted from HTML page structure could be further used, for instance, in sentiment detection.

Other possible use of content extraction would be in preparation of training sets for text-analysis algorithms. Many websites contain posts already categorized by some criterion. For example, product reviews site with opinions divided by emotional affection of the author could be mined and used as a teaching set for sentiment classifier. With this approach we could easily update the set if needed just by downloading new opinions. Such proceeding would assure the set will not become outdated, despite the rapid changes of linguistic forms used on the Internet. Emerging new words and phrases could be quickly incorporated into the set.

1.3. Related work

With the growing popularity of the Internet, more and more scholars are engaged in research on wrapper generators. There are two main approaches to the problem: semi-automatic wrapper induction and automatic wrapper generation.

Wrapper induction is the older way of tackling the problem. In those solutions, a manually labelled teaching set is used to derive extraction rules during supervised learning. There are a few notable works in this field. The first one is STALKER algorithm (Muslea, Minton & Knoblock, 1998), which generates formal data extraction rules for selected websites. It requires only few training examples and uses a set of disjunctive landmark automata organized in a hierarchy. The second often referenced technique is Boosted Wrapper Induction (Freitag & Kushmerick, 2000). It uses machine learning algorithm, improved by repeatedly applying it to the training set with different example weightings.

The requirement of manual work is a considerable disadvantage of the mentioned methods. Drawbacks would be especially visible in a long-term continuous data extraction (e.g. in social media monitoring). When the structure of the source changes, the initial set preparation and teaching process would have to be performed again. Due to these aspects further research has been made on automation of wrapper maintenance. The proposed machine learning approach (Lerman, Minton & Knoblock, 2003) is based on an idea of storing the correctly extracted data chunks. In case of a source template change, the training set is able to self-update the position of labels by searching for content matching previously acquired information.

Nevertheless, the shortcomings of wrapper induction motivated researchers to pursue more universal generation methods. The idea behind the automatic wrapper generation approach is to extract data by searching for patterns in the page content. Additional heuristics are used to help locate data series and data fields inside each data record. Of course, information from various domains has different characteristics, which makes it difficult to establish universal algorithms.

There are substantially more solutions in this category. ROADRUNNER (Crescenzi, Mecca & Merialdo, 2002) is based on an analysis of similarities in documents and creates a wrapper by inferring a generalized regular expression after analyzing multiple pages. IEPAD (Chang & Lui, 2001) takes into consideration that most of the websites with user generated content use only one template for displaying the data from one tuple and automatically identifies records by repeated pattern mining of HTML structures. More adaptable ODE (Su, Wang & Lochovsky, 2009) algorithm harnesses domain specific ontologies to identify data regions and to align and label the data values in records. DEPTA

(Zhai & Liu, 2005) renders analyzed pages and uses visual boundaries of data records together with string edit distance to locate data series.

There are also methods of statistical origin, which do not require any domain specific knowledge or heuristics. Hierarchical clustering algorithms could be used for data region discovery and segmentation (Papadakis et al., 2005). Another approach could be to analyse the source using Markov Logic Networks in order to efficiently perform distinction between parts of the document belonging to layout and those being data record (Satpal et al., 20110).

Some algorithms make use of the multi-level examination. This method takes into account the fact that often lists contain links to single pages, where each record is presented in detail (Lerman et al., 2004). Such knowledge improves record localization. However, this method has no use in the analysis of social media, where only few websites have special detailed page for each statement.

Recent approaches dedicated to extracting data from web forums include:

- Analysis of repetitions of both tag structures and visible text tokens (Li et al., 2009).
- Using specific domain constraints and utilizing them for data region segmentation (Song et al., 2010).
- Site-wide analysis of multiple pages in terms of fulfilling certain features (Yang et al., 2009). Markov model networks are used to build a probabilistic model for localising data records and their data fields.

Nevertheless, the field of wrapper generation is still open to new ideas. Despite a decade of research, many analyses show that aforementioned techniques are not sufficient (Liu, Grossman & Zhai, 2003; Weninger et al., 2011).

1.4. Motivation

The lack of efficient techniques for generic statement extraction was our main concern during collection of data from social media websites. Therefore we decided to design and implement a wrapper generation algorithm for the user generated content websites. The main goals were to achieve high precision and ensure complete automation of wrapper generation that would result in no manual maintenance work. We also wanted to establish high computing efficiency that would allow us to use the algorithm in distributed web crawling system and mine posts from approx. 5 thousands websites.

The last requirement for the algorithm was to preserve low level of false positives, i.e. to avoid creating wrapper for a page where no posts exists. Examples of such pages include list of topics on a web forum or list of news on an information portal. Apart from creating special filters for such cases, we also wanted to satisfy this requirement by being able to infer a generalized wrapper for a given site. Such site-specific wrapper, that would be a result of the analysis of multiple pages from that site, would further reduce extraction of improper posts.

2. Algorithm

2.1. Data regions and data records

In order to describe our algorithm we need to define the following notions:

Data field – one of the properties of a post, for instance its content or date of creation.

Data record – a node in the DOM tree of a webpage, which contains a single post with all of its data fields.

Data region – a node in the DOM tree, which contains multiple data records.

2.2. General idea of the algorithm

Our algorithm is based on the analysis of the document tree and proceeds by creating list of all potential candidate data regions, then filtering and scoring to select one data region that most probably contains posts from users. Afterwards, the final region is analyzed by various heuristics to locate data fields inside each data record. Moreover, we have developed a tree-matching technique for generalization of a wrapper, which allows us to create a site-specific wrapper in the form of an XPath expressions.

The overall procedure of the algorithm for generating a wrapper from a single webpage could be stated as follows:

1. Convert HTML mark-up of the document to a tree object model.
2. Create initial data regions set by searching for repetitive tag names.
3. Select final data region and data records by filtering using the following methods:
   (a) Tag Number Filter
   (b) Dummy Tree Matching Filter
   (c) Date Exists Filter
   (d) Score Filter.
4. Locate most common paths for data fields inside the data region:
   (a) Locate date of post path using Date Matching Location Finder
   (b) Locate content path using Edit Distance Location Finder.

The initial data region set creation and Dummy Tree Matching Filter are inspired by the methods proposed in the Tree Wrap algorithm (Hong & Fauzi, 2010). The main contributions of our algorithm are utilising the date matching for data region location, methods for selection of the best region and localising the data fields inside data records. Also, we developed the technique for wrapper generalisation. It uses multiple wrappers, generated for various pages from a given website, to create a single wrapper for that site.

Wrapper generalisation allows us to overcome the main limitation of our algorithm, which is the requirement that the analysed page contains at least three occurrences of posts. In order to extract information from pages containing one or two posts, we can use a generalised wrapper created on pages from the same site that contain three or more posts.

2.3. HTML mark-up to tree object model conversion

HTML4 is not an XML-compliant standard, in consequence requiring conversion to acquire a proper tree-based document model. The newer XHTML standard is theoretically compatible with XML. However, in reality many of the webpages fail the W3C mark-up validation test. This results in the need to clean up the HTML code to correct minor errors, e.g. unclosed tags, tag name misspellings, etc. Additionally, at this step we remove 34 irrelevant text-formatting tags. This causes flattening of the document tree model and facilitates creation of a wrapper.

2.4. Creation of initial data regions

The main observation is that data records are contained within nodes with equivalent tag names. Almost invariably those nodes are on corresponding level, having the same root node. Using these findings we create the initial candidate set by searching for repetitive tags on each possible level. Search begins at the very root node and recursively delves into each child. Every possible tag name that occurs three or more times as a child of the examined node is taken into account as a potential data region. At the end of this procedure we obtain a list of all candidate data regions.

2.5. Filtering data regions

The second step consists of applying various filters that remove irrelevant data regions and/or data records. After applying each filtering step, all data regions containing less than three data records are removed.

2.5.1. Tag Number Filter

This filter takes advantage of the observation that each post (i.e. data record) has relatively complex HTML code and it should contain at least three data fields (content, author, date). It counts the number of all nodes inside tree structure of each data record, and if it contains less than three nodes, the data record is removed from the data region.

2.5.2. Dummy Tree Matching Filter

In this step we take into consideration the fact that websites use templates for code generation. In consequence, each of the data records should follow similar tag structure. Only minor differences should be noted, due to the use of formatting tags in the content of the data record. The filter matches the tree structures of records and removes those with anomalies. Matching is performed in a "dummy" way, by counting the number of unique tags in each data record at the top level and with the recursion at all levels. If the difference exceeds the threshold, the record is removed.

2.5.3. Date Exists Filter

This phase derives from a conclusion that each of the data records must contain a date of post creation in some text format. During collection of 231 test websites, we were not able to find any website that would not display the dates of posts. However, checking whether the examined data record has a date field, turned out to be an exceptionally difficult task. Many of web forums engines allow to customize the date display format. This results in extremely wide range of possible date formats. Additional problem appeared because majority of our test cases were Polish web forums. We had to include specific matchers for Polish language that were able to recognize months in various linguistic forms. For instance, January can be written in Polish as "Styczeń", "Styczniu" or "Stycznia" depending on the day of the month. Moreover, we also had to implement recognition for various common cases:

- misconfiguration of the date display format, often resulting in doubled year ("17 lutego 2012, 2012 19:32") or AM/PM mark in 24-hour clock ("8 Jan 2009, 17:45 pm"),
- usage of the "<time>ago" patterns (e.g. "3 minutes 20 seconds ago", "1 year 7 days ago", etc.),
- displaying day and months numbers using Roman numerals ("22 II 2012, 17:53"),
- different combinations of punctuation marks ("20.Jun.2011 14:53"),
- character strings looking similar to dates ("Version 3.12.8").

The resulting date string recognition tool consists of 61 regular expression matchers, divided into three groups. Each group have different text preprocessing that removes characters irrelevant to the matcher group. We prepared a test suite containing 307 tests for various date formats, including 22 tests for existence of false positives.

2.5.4. Score Filter

The final step of filtering involves scoring resulting data regions. Only region with highest score is passed to the next phase of algorithm. This step is based on the following observations:

1. Regions containing posts are relatively large comparing to the whole page (text length multiplier).
2. User generated content is usually deeply embedded in HTML structure (HTML depth multiplier).

The scoring function equation is as following:

$$SCORE(dataregion) = length * L + depth * D \tag{1}$$

Where: *length* – text length of the data region; L – text length multiplier (constant); *depth* – level of depth of the data region in HTML tree structure; D – depth multiplier (constant). Exact parameters for L and D were found using genetic search algorithm. We had evaluated the fitness of candidates using the correctness of whole algorithm applied on our test set (see 3.1).

2.6. Locating fields inside data records

The previous stage of the algorithm gives us a list of parts of a website that may be statements – the data records within the final data region. From those records we extract final information – for each post we retrieve content and date. For each of the fields above we seek location that is common for all given records. A *location* is information on how to traverse document tree in order to reach the target node. Each location is a list of locating elements, where each element contains following information about tag to select:

1. name of the tag;
2. index of the tag, in case there are multiple tags with the same name;
3. alpha prefix of class attribute of the tag;
4. alpha prefix of id attribute of the tag.

Alpha prefix is the longest prefix containing only letters. Note that any found location can be represented as an XPath expression.

2.6.1. Content location

Upon the analysis HTML source we can make an observation that content is always in the same location, but the inner text of it is very variable. Therefore, we perform edit distance measurements between possible content locations with the idea, that the location with biggest differences between records would be our content location. More exactly, the algorithm works as follows:

1. For every data record we look for a node with longest text content. We save a generic XPath expression allowing to localise that node to the list of candidates.
2. After analysing all data records, we remove duplicated location expressions.
3. We calculate the score of each location expression. For a given expression, we try to apply it to every data record. The resulting series of possible post contents is examined using the Levenstein edit distance algorithm[1]

[1]Levenstein edit distance is equal to the minimum number of edits (deletions, insertions and substitutions) that have to be made to transform one string to another (Levensthein, 1966).

to measure differences between each adjacent post content. We save the sum of length-normalised edit distances as the score.
4. Each content location is examined for existence of hyperlinks. The purpose of this step is to reduce the number of false positives. For instance, a false positive occurs when a list of threads is wrongly classified as a list of posts. We observed that in such cases the content is mainly inside <a> tags. Therefore, we avoid them by applying the following filter. If the number of links divided by the number of words in possible content is greater than 0.04, then we reject such location[2].
5. The location expression that achieved the biggest score and passed the hyperlinks filter is returned as the common location of content nodes.

2.6.2. Date location

Every node in record which is short enough is matched against generic date recognition tool that was described earlier in the Date Exists Filter. Note that data record can contain many dates (i.e. when author joined forum, when post was written, date of last edition) but only the most recent one is returned. Like in the previous step, we find the most common location for the date field in data record and return it.

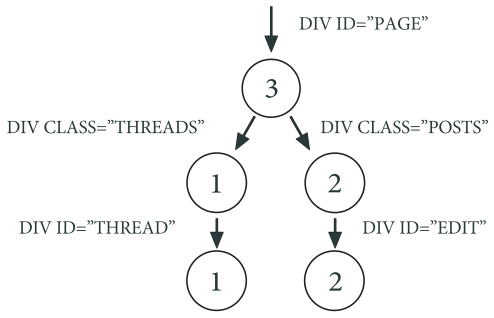
2.7. Wrapper generalisation

In order to generalize a wrapper, first we need to obtain wrappers generated with our algorithm for multiple pages. Afterwards the following generalisation procedure is applied:

1. For each page, apply the generated wrapper and return location for each found data record, together with content and date location inside that record.
2. Gather locations from documents into three lists: record locations, content locations, date locations.
3. Apply following procedure on each of the lists:
   (a) Create a shared tree of locations, using location elements as nodes.
   (b) Select the longest path from the shared tree root to the node which is used in at least 40% of locations.
   (c) Use nodes selected by that path as location elements to create shared location object.
   (d) Return shared location object as a generalized wrapper for that type of location.

[2]Value of the threshold was determined by probing possible values with step = 0.005

Figure 1. Example of a shared tree of location elements



2.7.1. Creating shared tree

The main idea for creating a shared tree of location elements (see Section 2.6)) is to find all possible locations while simultaneously computing similarities between them. The procedure for creating a tree is as follows:

```
1. For each location:
    (a) Set current node $u$ to tree root.
    (b) For each location element $e$:
        i. If current node $u$ has connection to node $n$ representing element
           $e$, then increase score on that connection, else create node representing
           element $e$ and create connection $c$ between them with score 1.
        ii. Set current node $u$ to node $n$.
```

Fig. 1 presents an example of a shared tree for following three locations:

- /div[@id="page"]/div[@class="posts"]/div[starts-with(@id, "edit")]
- /div[@id="page"]/div[@class="posts"]/div[starts-with(@id, "edit")]
- /div[@id="page"]/div[@class="threads"]/div[starts-with(@id, "thread")]

2.7.2. Selecting longest path

In order to select the shared location the following method is used:

```
1. Set current node $u$ to tree root.
2. While current node $u$ has any children that are shared in at least 40% of
   locations, do:
    (a) Find child $c$ of current node $u$ that has the biggest score.
    (b) Add location element $e$ represented by $c$ to the shared location.
    (c) Set current node $u$ to $c$.
```

Table 1. Sizes of the test sets

Test set name	Size
social media websites	248
false positives	95
all	343

Table 2. Web forum engines statistics

Engine name	Custom	phpBB	IP.Board	vBulletin	Other	SMF
Count	97	96	56	46	34	14

3. Experiments

3.1. The test set

We have manually prepared a test set containing 343 HTML pages in Polish and English language from web forums and other social media websites. All pages contain in total 3056 posts. The test set was divided into two groups (see Table 1).

First test set included only HTML pages that contained posts and was called "social media websites". Test cases came from 248 different websites that differ in HTML structure and forum engine. For statistics of the forum engines appearing in the test pages see Table 2.

The second test set called "false positives" was composed only of HTML pages that contained no posts, but had a document structure that may deceive the algorithm (repetitive nodes, dates inside text, etc.). Test cases contain mainly lists of threads and topics from web forums as well as front pages of websites.

3.2. Methodology

3.2.1. Wrapper generation

For every web page we created a meta document describing expected extraction result. Every meta document includes two "golden posts": one from the beginning of a web page and the other from the end. There are only two posts, because if a page is extracted correctly, then all posts are correct and none otherwise. Nevertheless, there are some minor errors which are acceptable for us. Those errors are:

1. Difference in expected and actual post count by one – usually the first, last or post that contains advertisement.

2. No hour and minute in extracted date.
3. Extracted post does not contain the whole actual post because of complexity (quotations or other blocks).

Therefore, we created two test suites - one which accepts the small errors (tolerant test) and second one that does not allow them (strict test). There are also two cases that partially test our algorithm. First one is the test of extracting records. It checks initial data region creation and filtering of data records. Second test focuses only on locating fields inside data records. Both of these tests are strict tests and do not allow any errors. We have computed the following widely used indicators:

Recall – number of correctly extracted documents/number of expected documents.

Precision – number of correctly extracted documents/number of extracted documents.

F-measure – 2 * (precision * recall)/(precision + recall).

Correctness – number of passed test cases/all test cases.

3.2.2. Comparison to other state-of-the-art algorithms

In order to compare our algorithm (named "Heuristic Wrapper Generator") with the state-of-the-art approaches we used the Structured Data Extractor (SDE)[3]. It is a publicly available Java implementation of the DEPTA algorithm (Zhai & Liu, 2005), though there are some minor differences in the tree-matching technique[4]. However, it turned out that this implementation has errors that arise in certain tests. Therefore, we limited the test suite to 181 tests that had executed properly without throwing any exceptions. The same indicators as in the previous tests were used. They were computed only for the the first phase, i.e. extracting data records without locating data fields inside records, as the implementation does not label the data fields.

3.2.3. Wrapper generalization

We have also prepared the test suite for checking correctness of the wrapper generalization algorithm. After randomly selecting 20 web forums from our previous test, we have downloaded 200 web pages for each web forum. Consequently, we applied the wrapper generation algorithm to each web page, including also those pages that could return false positives, e.g. lists of topics. Accuracy was measured by recall, i.e. number of correctly extracted documents divided by number of expected documents. We retrieved posts using generalized wrapper and compared them with posts extracted with manually prepared wrapper based on XPath expressions. All posts were compared on content and date of creation.

[3]http://seagatesoft.blogspot.com/2012/05/structured-data-extractor.html
[4]http://seagatesoft.blogspot.com/2012/07/differences-between-sde-and-depta.html

3.3. Test results

For detailed results of the algorithm on the test set containing social media websites see Table 3. As we can see, in tolerant test our algorithm achieved 6.05% better correctness. This inequality comes mainly from the difference between the expected and actual post count of one. This is often caused by advertisements, which are hardly distinguishable, because they usually contain all characteristics of a true post.

We have also tested our algorithm on the test set that includes pages, which do not contain any proper posts (test for false positives). See Table 4 for details. It should be noted that the algorithm performs on the same level of correctness as the test without false positives.

In order to ensure that all steps of the algorithm are necessary, we have analysed the performance of each data region filter. In Fig. 2 the average number of data regions at each step can be found. We can notice that Tag Number Filter and Date Exists Filter significantly contribute to the performance of the algorithm.

However, our algorithm is not working correctly in some test cases. As we can see in Table 5 the main reason for test failures is locating fields inside data records (6.45%), which means that the algorithm had found HTML node containing wanted post but could not obtain location of data fields or did it improperly. The next most common reason for failures is that the expected region was never included in the initial set of candidate regions. This is due to some website posts having HTML structure of a tree rather than list, i.e. replies are embedded inside the posts. Another reason for failures is that the expected data record was removed, because the HTML node containing it had too simple markup (Tag Number Filter 2.42%). The reason of failures for Dummy Tree Matching are in most cases advertisements, which are often automatically inserted between user posts on message boards. As far as Score Filter is concerned, it does not handle sites where advertisements are relatively longer than the user generated content. The rarest cause for test failure is detecting date inside data records. The vast majority of websites use standard time formats, which can be easily detected and parsed.

The aforementioned errors are effectively reduced by wrapper generalization. As we can see in Table 6, wrapper performance after generalization is very high, with 13 of 20 test cases passing with 100% correctness. Only two test cases achieve recall below 95%.

In the comparison with Structured Data Extractor our algorithm notes much better performance. Results in the Table 7 show significant difference in correctness, with our algorithm noting 93% compared to 24% of the SDE algorithm.

Table 3. Results of the application of the algorithm on the test set “social media websites”

Test type	Strict test	Tolerant test	Extracting data records	Locating fields inside data records
Recall	73.06%	78.57%	90.38%	74.37%
Precision	79.42%	85.40%	98.93%	76.76%
F-measure	0.761	0.818	0.945	0.755
Correctness	76.61%	82.66%	90.38%	82.26%

Table 4. Results of the application of the algorithm on the test sets “false positives” and “all”

Test type	“false positives”	Strict test for “all”	Tolerant test for “all”
Test cases	95	343	343
Recall	-	73.06%	78.56%
Precision	-	73.75%	79.32%
F-measure	-	0.734	0.789
Correctness	77.89%	76.96%	81.34%

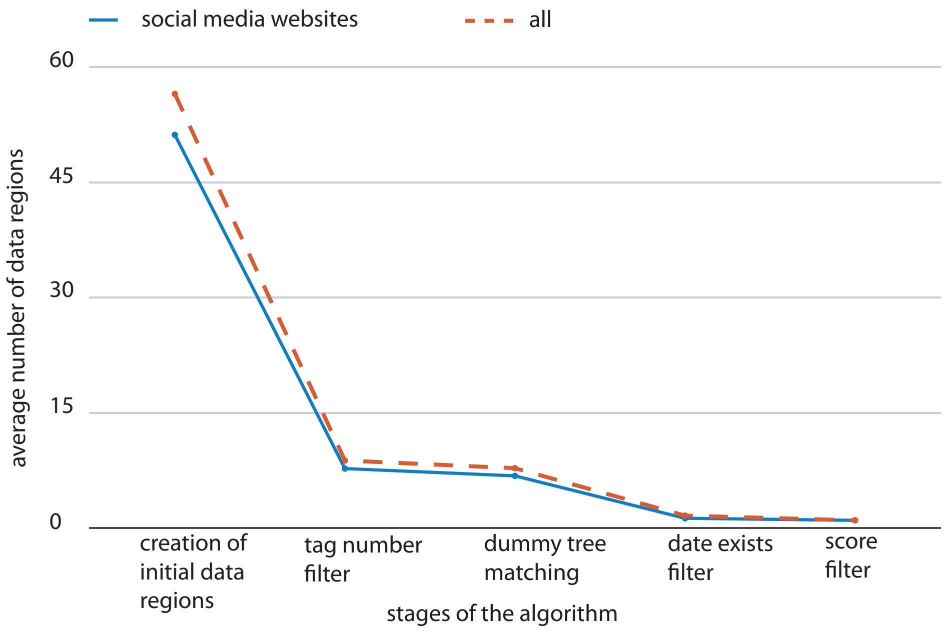
Table 5. Stages on which the algorithm has failed

Algorithm stage	Number of caused test failures
Locating fields inside data records	16 (6.45%)
Creation of initial data regions	9 (3.63%)
Tag number filter	6 (2.42%)
Dummy tree matching	5 (2.02%)
Score filter	5 (2.02%)
Date exists filter	2 (0.81%)

Table 6. Correctness of wrapper generalization

Website name	Number of correctly extracted posts	Recall
forum.pclab.pl	259/259	100.00%
forum.komputerswiat.pl	277/287	96.52%
xboxforum.pl	309/310	99.68%
forum.android.com.pl	286/300	95.33%
forum.benchmark.pl	367/367	100.00%
forum.pccentre.pl	156/157	99.36%
www.forum-bankowe.pl	384/392	97.96%
www.strefacaraudio.pl	244/244	100.00%
forumkomputerowe.pl	198/198	100.00%
forum.ipfon.pl	242/336	72.02%
forum.motocyklistow.pl	168/168	100.00%
forum.opel24.com	252/252	100.00%
forumprawne.org	153/153	100.00%
klubkm.pl	360/360	100.00%
www.audiostereo.pl	232/232	100.00%
www.forumsamochodowe.pl	219/219	100.00%
www.qell.pl	497/497	100.00%
www.forumrowerowe.org	436/436	100.00%
www.psxextreme.info	461/461	100.00%
www.zerom2i.fora.pl	10/16	62.50%
Overall	5510/5628	97.90%

Figure 2. Average number of data regions after each step of the algorithm.



4. Summary

We find our results very promising on the account that our algorithm requires only one webpage to generate a wrapper and is fully automatic. Comparison with the SDE algorithm shows much better performance of our algorithm, indicating that universal approaches like SDE do not perform well in specialised extraction tasks.

Our plan to address imperfections of the algorithm by broadening the analysis to multiple pages from each website proved successful. Such approach eliminated the errors resulting from single-page anomalies.

We believe that with the small error tolerance, the correctness of the algorithm is sufficient to allow integration in production systems. Therefore, we have run our wrapper generation algorithm on a 4-server crawling cluster. Each of the computers was based on an Intel Core i5 2.66GHz processor with 16GB RAM. The 24-hour test on 2197 social media sites allowed us to generate an average of 145.16 wrappers per second, in an environment where connection bandwidth was not the bottleneck.

In summary, we evaluate the algorithm as very useful. We are currently using it to extract posts from forums on capital markets. We intend to analyse the sentiment of the statements and seek correlation with stock market behaviour.

Table 7. Comparison of our Heuristic Wrapper Generator with the SDE algorithm

Algorithm	Heuristic Wrapper Generator	Structured Data Extractor
Recall	92.94%	25.07%
Precision	98.76%	37.71%
F-measure	0.960	0.301
Correctness	92.94%	24.41%

4.1. Acknowledgments

The work presented hereby has been done as a part of Thivera research project, supported by the Foundation for Polish Science [grant Ventures/2011-7/1]. We would like to thank dr Wojciech Waloszek for his invaluable assistance.

References

Chang, C. & Lui, S. (2001) IEPAD: information extraction based on pattern discovery. In: *Proceedings of the 10th International Conference on world Wide Web.* ACM, New York, NY, USA, 681–668.

Cong, G., Wang, L., Lin, C., Song, Y. & Sun, Y. (2008) Finding question-answer pairs from online forums. In: *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval.* ACM, New York, NY, USA, 467–474.

Crescenzi, V., Mecca, G. & Merialdo, P. (2002) RoadRunner: automatic data extraction from data-intensive web sites. In: *Proceedings of the 2002 ACM SIGMOD international conference on Management of data.* ACM, New York, NY, USA, 624–624.

Freitag, D. & Kushmerick, N. (2000) Boosted wrapper induction. In: *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence.* AAAI Press, Menlo Park, CA, USA, 577–583.

Glance, N., Hurst, M., Nigam, K., Siegler, M., Stockton, R. & Tomo–kiyo, T. (2005)Deriving marketing intelligence from online discussion. In: *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining.* ACM, New York, NY, USA, 419–428.

Hong, J. & Fauzi, F. (2010) TreeWrap-Data Extraction Using Tree Matching Algorithm. In: *Majlesi Journal of Electrical Engineering.* Islamic Azad University, Majlesi, 4 (2), 43–55.

Kim, P. (2006, Q3) The forrester wave: Brand monitoring (white paper). Forrester Wave, Cambridge, MA, USA.

Kushmerick, N. (1997) Wrapper induction for information extraction (doctoral dissertation). University of Washington.

Lerman, K., Getoor, L., Minton, S. & Knoblock, C. (2004) Using the structure of Web sites for automatic segmentation of tables. In: *Proceedings of the 2004 ACM SIGMOD international conference on Management of data.* ACM, New York, NY, USA, 119–130.

Lerman, K., Minton, S. & Knoblock, C. (2003) Wrapper maintenance: a machine learning approach. In: *Journal of Artificial Intelligence Research.* AI Access Foundation, El Segundo, CA, USA, 18 (1), 149–181.

Levenshtein, V. (1966) Binary codes capable of correcting deletions, insertions and reversals. In: *Soviet physics-doklady.* American Institute of Physics, New York, NY, USA, 10 (8), 707-710.

Li, S., Tang, L., Hu, J. & Chen, Z. (2009) Automatic Data Extraction from Web Discussion Forums. In: *Proceedings of the 2009 Fourth International Conference on Frontier of Computer Science and Technology.* IEEE Computer Society Washington, DC, USA, 219–225.

Liu, B., Grossman, R. & Zhai, Y. (2003) Mining data records in Web pages. In: *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, New York, NY, USA, 601–606.

Muslea, I., Minton, S. & Knoblock, C. (1998) Stalker: Learning extraction rules for semistructured, web-based information sources. In: *Proceedings of AAAI-98 Workshop on AI and Information Integration.* AAAI Press, Menlo Park, CA, USA, 74–81.

Pang, B. & Lee, L. (2008) Opinion mining and sentiment analysis. Now Publishers Inc., Boston-Delft.

Papadakis, N., Skoutas, D., Raftopoulos, K. & Varvarigou, T. (2005) An automatic web wrapper for extracting information from web sources, using clustering techniques. In: *Proceedings of the 2005 Symposium on Applications and the Internet.* IEEE Computer Society Washington, DC, USA, 24–30.

Satpal, S., Bhadra, S., Sellamanickam, S., Rastogi, R. & Sen, P. (2011) Web information extraction using markov logic networks. In: *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, New York, NY, USA, 1406–1414.