Web Content Extraction – a Meta-Analysis of its Past and Thoughts on its Future

Tim Weninger
University of Notre Dame
Notre Dame, Indiana, USA
tweninge@nd.edu

Rodrigo Palacios
California State University
Fresno, California, USA
rodpl91@gmail.com

Valter Crescenzi
Università Roma Tre
Dipartimento di Ingegneria,
Rome, Italy
crescenz@dia.uniroma3.it

Thomas Gottron
Institute for Web Science and Technologies
University of Koblenz-Landau, Germany
gottron@uni-koblenz.de

Paolo Merialdo
Università Roma Tre
Dipartimento di Ingegneria,
Rome, Italy
merialdo@dia.uniroma3.it

ABSTRACT

In this paper, we present a meta-analysis of several Web content extraction algorithms, and make recommendations for the future of content extraction on the Web. First, we find that nearly all Web content extractors do not consider a very large, and growing, portion of modern Web pages. Second, it is well understood that wrapper induction extractors tend to break as the Web changes; ; heuristic/feature engineering extractors were thought to be immune to a Web site's evolution, but we find that this is not the case: heuristic content extractor performance also tends to degrade over time due to the evolution of Web site forms and practices. We conclude with recommendations for future work that address these and other findings.

1. INTRODUCTION

The field of content extraction, within the larger pervue of data mining and information retrieval, is primarily concerned with the identification of the main text of a document, such as a Web page or Web site. The principle argument is that tools that make use of Web page data, *e.g.*, search engines, mobile devices, various analytical tools, demonstrate poor performance due to noise introduced by text not-related to the main content [11; 23].

In response the field of content extraction has developed methods that extract the main content from a given Web page or set of Web pages, *i.e.*, a Web site [20; 29]. Frequently, these content extraction methods are based on pattern mining and the construction of well-crafted rules. In other cases, content extractors learn the general skeleton of a Web page by examining multiple Web pages in a Web site [18; 1; 6; 7]. These two classes of content extractors are referred to as *heuristic* and *wrapper induction* respectively; and each class of algorithms have their own merits and disadvantages. Generally speaking, wrapper induction methods are more accurate than heuristic approaches, but require some amount of training data in order to initially induce an appropriate wrapper. Conversely, heuristic approaches are able to function without an induction step, but are generally less accurate.

The main criticism of content extraction via wrapper induction is that the learned rules are often brittle and are unable to cope with even minor changes to a Web pages' template [12]. When a Web site modifies its template, as they often do, the learned wrappers need to be refreshed by re-computing the expensive induction step. Certain improvements in wrapper induction attempt to induce extraction rules that are more robust to minor changes [9; 8; 25], but the more robust rules only delay the inevitable [5].

Heuristic approaches are often criticised for their lack of generality. That is, heuristics that may work on a certain type of Web site, say a news agency, are often ill suited for business Web sites or message boards, etc. Most approaches also ignore the vast majority of the Web pages that dynamically download or incorporate content via external reference calls during the rendering process, *e.g.*, CSS, JavaScript, images.

The goal of this paper is not to survey the whole of content extraction, so we resist the temptation to verbosely compare and contrast the numerous published methods. Rather, in this paper we make a frank assessment on the state of the field, provide an analysis of content extraction effectiveness over time, and make recommendations for the future of content extraction.

In this paper we make three main contributions:

1. We define the vectors of change in the function and presentation of content on the Web,
2. We examine the state of content extraction with respect to the ever changing Web, and
3. We perform a temporal evaluation on various content extractors

Finally, we call for a change in the direction of content extraction research and development.

The evolution of Web practices is the central to the theme of this paper. A scientific discipline ought to strive to have some invariance in the results over time. Of course, as technology changes, our study of it must also change as well. With this in mind, one way to determine the success of a model is to measure its stability or durability as the input changes over time.

To that end, we present the results of a case study that compares content extraction algorithms, both old and new, on an evolving

Web site
news.bbc.co.uk
cnn.com
news.yahoo.com
thenation.com
latimes.com
entertainment.msn.com
foxnews.com
forbes.com
nymag.com
esquire.com

Table 1: Dataset used in case study. 25 Web pages crawled from each Web site per lustrum (5-year period), over 4 lustra and 10 Web sites totals 1,000 Web pages.

dataset. The goal is to identify which measures, if any, are invariant to the evolution of Web practices.

To that end, we collected a dataset of 1000 Web pages from 10 different domains, listed in Table 1, where each domain has a set of pages from years 2000, 2005, 2010, and 2015. There are 25 HTML documents per lustrum (*i.e.*., 5-year period), for a total of 100 documents per Web site. The documents were automatically and manually gathered from two types of sources: archives[1] and the original websites themselves for the 2015 lustrum.

We review the evolution that has occurred in Web content delivery and extraction, referring explicitly to recent changes that undermine the effectiveness of exiting content extractors. To show this explicitly we perform a large case study wherein we compare the performance over time of several content extraction algorithms. Based on our findings we call for a change in content extraction research and make recommendations for future work.

2. EVOLVING WEB PRACTICES

We begin with the observation that the content delivery on the Web has changed dramatically since it was first conceived. The case for content extraction is centered around the philosophy that HTML is a markup language that describes how a Web page ought to *look*, rather than what a Web page contains. Here, the classic form versus function debate is manifest. Yet, in recent years the Web has seen a simultaneous marriage and divorce of form and function with the massive adoption of scripting languages like JavaScript and with the finalization of HTML5.

In this section we argue that because Web technologies have changed, the way we perform and evaluate content extraction must also change.

2.1 Evolution of Form and Function

JavaScript. Nearly all content extraction algorithms operate by downloading the HTML of the Web page(s) under consideration, and only the HTML. In many cases, Web pages refer directly or indirectly to dozens of client side scripts, *i.e.*, JavaScript files, that may be executed at load-time. Most of the time content extractors do not even bother to download referenced scripts even though JavaScript functions can (and frequently do) completely modify the DOM and content of the downloaded HTML. Indeed, most of the spam and advertisements that content extraction technologies explicitly claim to catch are loaded via JavaScript and are therefore not part of most content extraction testbeds.

CSS. Style sheets pose a problem similar in nature to JavaScript in that structural changes to the displayed content on a Web site are frequently performed by instructions embedded in cascading style sheets. Although CSS instructions are not as expressive as JavaScript functions – they were built for different purposes – the omission of a style sheet often severely affects the rendering of a Web page.

Furthermore, many of the content extractors described earlier rely on formatting hints that live within HTML in order to perform effective extraction. Unfortunately, the ubiquitous use of CSS removes many of the HTML hints that extractors depend upon. Using CSS, it is certainly possible that a complex Web site is made entirely of `div`-tags.

HTML5. The new markup standards introduced by HTML5 include many new tags, including `main`, `article`, `header`, etc., meant to specify the semantic meaning of content. Widespread adoption of HTML5 is in progress, so it is unclear whether and how the new markup languages will be used or what the negative side effects will be, if any.

The semantic tags in HTML5 are actually a severe departure form the original intent of HTML. That is, HTML4 was originally only meant to be a markup for the visual structure of the Web page, not a description language. Indeed the general lack of semantic tags is one of the main reasons why content extraction algorithms were created in the first place.

Further addition of semantics into HTML markup is provided by the `schema.org` project. Schema.org is a collaboration among the major Web search providers to provide a unified description language that can be embedded into HTML4/5 tag attributes. Web site developers can use these tags to encode whatthat certain HTML data represents, for example, a Person-`itemtype`, which may have a name-`itemprop`, which can then be used by search engines and other Web-services to built intelligent analytics tools. Other efforts to encode semantic meaning in HTML can be found in the `Microformats.org` project, the Resource Description Framework in Attributes (RDFa) extension to HTML5, and others.

	itmscp	itmtp	itmprp	sctn	artcl
Mean	162.2	157.8	899.0	261.0	403.4
Median	65.5	54.5	374.5	25	166.5

Table 3: Mean and Median number of occurrences of semantic tags from `schema.org`: `itemscope`, `itemtype` and `itemprop` tags, and from HTML5: `article` and `section` found in 2015-subset of the dataset. Semantic tags are only found in dataset from 2015.

Table 3 shows the mean and median number of Schema.org and HTML5 semantic tags in our 2015 dataset. We find that 9 out of 10 Web sites we crawled had adopted the Schema.org tagging system, and that 9 out of 10 Web sites had adopted the `section` and `article` tags from HTML5 (8/10 adopted both Schema.org and HTML5).

The advent and widespread adoption of HTML5 and Schema.org decreases the need for many extraction tools because the content or data is explicitly marked and described in HTML.

AJAX. Often, modern Web pages are delivered to the client without the content at all. Instead, the content is delivered in a separate JSON or XML message via AJAX. These are not rare cases, as of April 2015, Web Technologies research finds that AJAX is used within 67% of all Web sites[2]. Thus, it is conceivable that the vast

[1] WayBack Machine - http://archive.org/web/

[2] `http://w3techs.com/technologies/overview/`

Figure 1: The Web page of `http://www.kdd.org/kdd2015/` fully rendered in a modern Web browser (Left). Web page with JavaScript disabled (Middle). Downloaded Web page HTML, statically rendered without any external content (Right). Most extractors operate on the Web page on the right.

	src	link	iframe	script	js	jquery	css	size
2000	37.092	1.152	0.388	7.600	6.588	0.908	1.828	39,121.90
2005	61.812	2.528	0.408	21.200	14.280	0.944	2.612	52,633.82
2010	57.976	10.104	1.408	44.044	24.096	1.000	10.468	81,033.89
2015	49.396	18.256	10.032	40.652	37.052	0.620	11.692	174,801.64

Table 2: The mean-average occurrences of certain HTML tags and attributes that represent ancillary source files in our dataset of 1,000 news Web pages over 4 equal sized lustra (5-year periods). The use of external content and client-side scripting has been growing quickly and steadily.

majority of content extractors over estimate their effectiveness in 67% of the cases, because a large portion of the final, visually-rendered Web page is not actually present in the HTML file. .

In fact,An observation which supports this hypothesis is, that in our experiments we find that the most frequent last-word found byin many content extractors on NY Times articles is “loading...”

Table 2 shows the mean-average number of occurrences of certain HTML tags and attributes that represent ancillary source files in our dataset of 1,000 Web pages. In this table, `src` refers to the occurrence of the common tag attribute which can refer to a wide range of file types. `link` refers to the occurrence of the `<link>` HTML tag which frequently (although not necessarily) references external CSS files. `iframe` refers to the occurrence of the HTML tag which is used to embed another HTML document into the current HTML document. `script` refers to the occurrence of the HTML tag which is used to denote a client-side script such as (but not necessarily) JavaScript. `js` refers to the occurrences of externally referenced JavaScript files; `css` similarly refers to the occurrences of externally referenced CSS files. The `jquery` column shows the percentage of Web pages that employ AJAX via the jQuery library; alternative AJAX libraries were found but their occurrence rates were very small.

In many ways the above observations show that the Web is trending towards a further decoupling of form from content: JavaScript decouples the rendered DOM from the downloaded HTML, CSS similarly separates the final presentation from the downloaded HTML, and AJAX allows for the HTML and extractable content to be separate files entirely. Yet, despite these trends, most content extraction methodologies rely on extractions from statically downloaded HTML files.

An example of why this should be considered a bad practice is highlighted in Figure 1 where the Web page `http://kdd.org/kdd2015` is shown rendered in a browser (at left), rendered without JavaScript (center), and rendered with only the static HTML document (at right). The information conveyed to the end user is presented in its complete form in the rendered version; thus, content extractors should strive to operate within the fully rendered document (at left), instead of the HTML-only extraction as is the current practice (at right).

`javascript_library/all`. Accessed May 6, 2015.

2.2 Keeping Pace with the Changing Web

Web presentation has evolved in remarkable ways in a very short time period. Content Extraction algorithms have attempted to keep pace with evolving Web practices, but many content extraction algorithms quickly become obsolete.

Counter-intuitively, it seems that as although the number of Web sites has increased, the variety of presentation styles has actually decreased. For a variety of reasons, most Web pages within the same Web site look strikingly similar. Marketing and brand-management often dictate that a Web site maintains style distinct from competitors, but are similar to other pages in the same Web site.

Wrapper Induction. The self-similarity of pages in a Web site stem from the fact that the vast majority of Web sites use scripts to generate Web page content retrieved from backend databases. Because of the structural similarity of Web pages within the same Web site, it is possible to reverse engineer the page generation process to find and remove the Web site’s skeleton, leaving only the content remaining [18; 1; 6; 7].

A wrapper is induced on one Web site at a time and typically needs only a handful of labelled examples. Once trained the learned wrapper can extract information at near-perfect levels of accuracy. Unfortunately, the wrapper induction techniques assume that the Web site template does not change. Even the smallest of tweaks to a Web site’s template or the database schema breaks the induced wrapper and requires retraining. Attempts to learn robust wrappers, which are immune to minor changes in the Web page template have beenshown somewhat successful, but even the most robust wrapper rules eventually break [8; 12].

Heuristics and Feature Engineering.
Rather than learning rigid rules for content extraction, other works have focused on identifying certain heuristics as a signal for content extraction. The variety of the different heuristics is impressive, and the statistical models learned through a combination of various features may, in many cases, perform comparable to extractors based on wrapper induction. Rather than learning rigid rules for content extraction, other works have focused on identifying certain heuristics as a signal for content extraction. The variety of the different heuristics is impressive, and the statistical models learned through a combination of various features may, in many cases, perform comparable to extractors based on wrapper induction.

Each methodology and algorithm was invented at a different time in the evolution of the Web and looked at different aspects of the Web content. From the myriad of options we selected 11 algorithms from different time periods. They are listed in Table 4.

Algorithm		Year
Body Text Extractor (BTE)	[11]	2001
Largest Size Increase (LSI)	[16]	2001
Document Slope Curve (DSC)	[27]	2002
Link Quota Filter	[21]	2005
K-Feature Extractor (KFE)	[10]	2005
Advanced DSC (ADSC)	[13]	2007
Content Code Blurring (CCB)	[14]	2008
RoadRunner* (RR)	[7]	2008
Content Extraction via Tag Ratios (CETR)	[28]	2010
BoilerPipe	[17]	2010
Eatiht	[24]	2015

Table 4: Content extraction algorithms, with their citation and publication date. *RoadRunner is a wrapper induction algorithm; all others are heuristic methods.

Each algorithm, heuristic, model or methodology is predicated on the form and function of the Web at the time of its development. Each was evaluated similarly on the state of the Web that existed at the time, presumably, just before publication. Furthermore, each algorithm does not consider JavaScript, CSS, or AJAX changes to the Web page, therefore the majority of the Web page may not actually be present for extraction, as is the case in Figure 1.

3. CASE STUDY

We present the results of a case study that compares content extraction algorithms, both old and new, on an evolving dataset. The goal is to test the performance variability of content extractors over time as Web sites evolve. So, for each Web page of each lustrum of each Web site, a gold-standard dataset was created manually by the second author. Each Web content extractor attempted to extract the main content from the Web page.

For the first seven content extractors in Table 4, we used the implementation from the CombineE System [13]. The Eatiht, BoilerPipe and CETR implementations are all available online. BoilerPipe provides a standard implementation as well as an article extractor (AE), Sentence extractor (Sen), an extractor trained on data from KrdWrd-Canola corpus[3], and two "number of words" extractors: a decision tree induced extractor (W) and a decision tree induced extractor manually tuned to at least 15 words per content area (15W). CETR has a default algorithm as well as a threshold option based on the standard deviation of the tag ratios (Th), and a 1 dimension clustering option (1D). See the respective papers for details.

An attempt was made to induce wrappers using the Roadrunner wrapper induction system [7], which was successful on each set of 25 Web pages, but performed very poorly on the proceeding lustrum. Wrapper-breakage is a well known problem for wrapper induction techniques [8; 12]. A five-year window is too long for any wrapper to continue to be effective. Thus Roadrunner had to be trained and evaluated slightly differently. In this case we manually identified Web pages that have very similar HTML structure and learned a wrapper on those few pages. In most cases 90-95% of the Web pages in a single domain could be used to generate a wrapper, but in 2 Web sites only about half of the Web pages were found to have the same style and were useful for training. We used the induced wrapper to extract the content from the Web pages on which it was trained.

We emphasize that our methodology follows that of most content extraction methodologies. Namely, we download the raw HTML of the Web page and perform content extraction on *only* that static HTML. We further emphasize that this ignores a very large portion of the overall rendered Web page – renderings that are increasingly reliant on external sources for content and form via AJAX, stylesheets, iframes, etc. The disadvantages of this methodology are clear, but we are beholden to them because the existing extractors require only static HTML.

3.0.1 Evaluation

We employ standard content extraction metrics to compare the performance of different methods. Precision, recall and F_1-scores are calculated by comparing the results/output of each methods to a hand-labeled gold standard. The F_1F1-scores are computed as usual and all results are calculated by averaging each of the metrics over all examples.

The main criticism of these metrics is that they are likely to be inflated. This is because every word in a document is considered to be distinct even if two words are lexically the same. This makes it impossible to align words with the original page and therefore forces us to treat the hand labeled content and automatically extracted content as a bag of words, *e.g.*i.e., where two words are considered the same if they are lexically the same. The bag of words measurement is more lenient and as a result scores may be inflated.

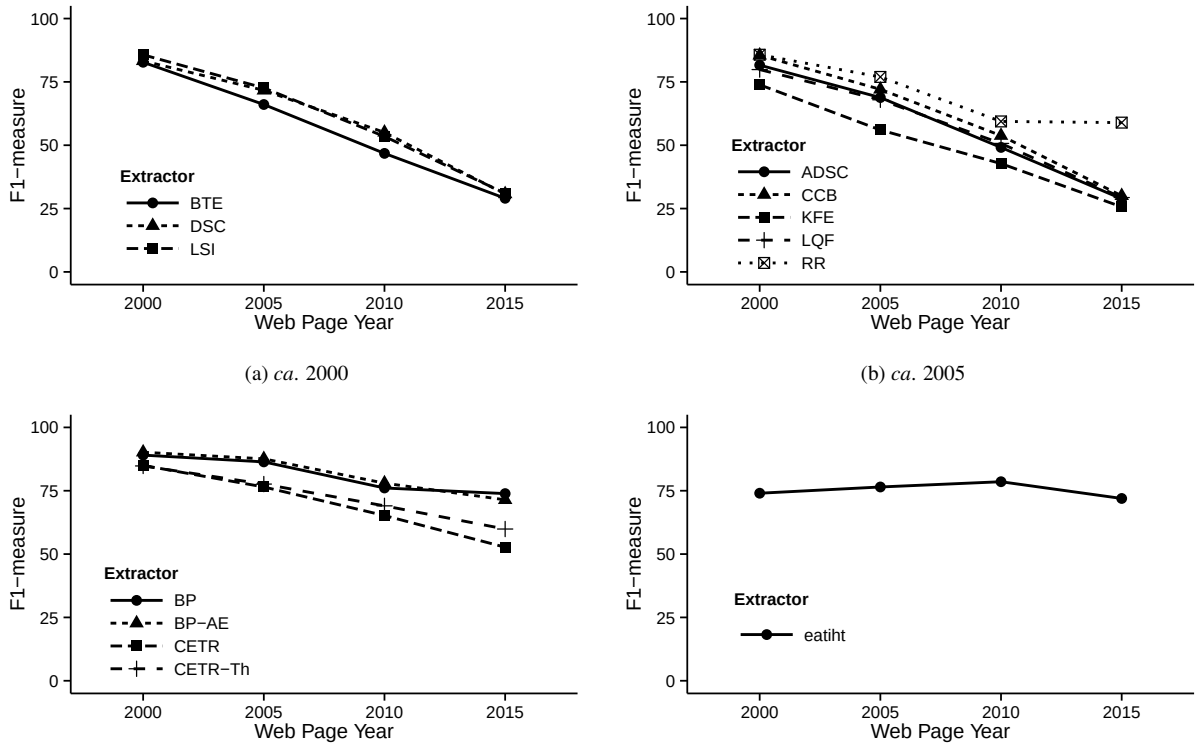
The CleanEval competition has a hand-labeled gold standard as well from a shared list of 684 English Web pages and 653 Chinese Web pages downloaded in 2006 by "[collecting] URLs returned by making queries to Google, which consisted of four words frequent in an individual language"[22]. CleanEval uses a different approach when computing extraction performance. Their scoring method is based on a word-at-a-time version of the Levenshtein distance between the extraction algorithm and the gold standard divided by the alignment length.

3.1 Results

First, we begin with a straightforward analysis of the results of each algorithm on the dataset. Figure 2a–2d shows the F_1-measure for each lustrum, *i.e.*, each 5-year time period, organized by extractor cohort. For example, the BTE-extractor was published in 2001, and is therefore part of the *ca*. 2000 cohort of extractors; it's performance is illustrated in Figure 2a. The eatiht-extractor was published in 2015 and is therefore part of the *ca*. 2015 cohort of extractors, and is illustrated in Figure 2d.

The shape of the performance curves in Figure 2a–2d over time exactly demonstrate the primary thesis of this paper: extractors quickly become obsolete.

[3] https://krdwrd.org/trac/raw-attachment/wiki/Corpora/Canola/CANOLA.pdf



(a) *ca.* 2000

(b) *ca.* 2005

(c) *ca.* 2010

(d) *ca.* 2015

Figure 2: F_1-measure for various extractor cohorts by lustrum (5-year period).

Indeed, Figure 3 averages the F_1-measure of each cohort and plots their aggregate performance together. We can clearly see that all of the extractor cohorts begin at approximately the same performance on Web page data from the year 2000, but the performance quickly falls as the form and function of the Web pages change. As a naive baseline, we also measure the results if all non-HTML text was extracted and treated as content; in this case, the F_1-measure is buoyed by the perfect recall score, but the precision and accuracies are bad as expected.

2015-extractors are most invariant to changes in the Web because the developers likely created the extractor knowing the state of the Web in 2015 and with an understanding of the history of the Web. 2010-extractors perform well on data from 2010 and prior, but were unable to adapt to unforeseen changes that appeared in 2015. Similarly extractors from 2005 performed well on data from 2005 and prior, but did not predict Web changes and quickly became obsolete.

The F_1-measure is arguably the best single performance metric to analyze this type of data, however, individual precision, recall and accuracy considerations may be important to various applications. The raw scores are listed in Table 5.

We find that extractors from 2000 and 2005 have a steep downward trend and extractors from 2010 also has a downward trend, although not as steep. Only the 2015 extractor performs steadily. These results indicate that changes Web design and implementation has adversely affected content extraction tools.

The semantic tags found in new Web standards like HTML5 may be one solution to the falling extractor performance. Table 6 demonstrates surprisingly good extraction performance by extracting only (and all of) the text inside the `article` tag from the 2015 lustrum

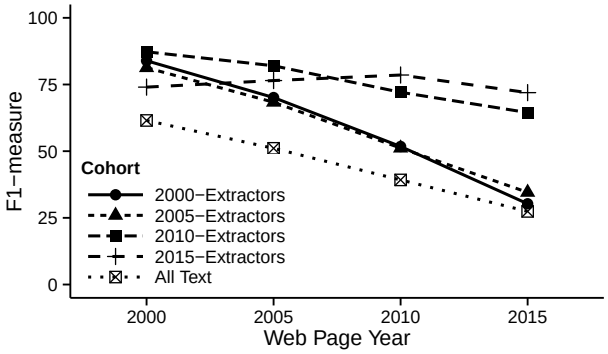


Figure 3: Mean average F_1 measure per cohort over each lustrum.

as the articles content. Compared to the results from the complex algorithms shown in Table 5 the simple HTML5 extraction rule shows reasonable results with very little effort.

	Precision	Recall	Accuracy
Mean	57.3	67.3	82.4
Median	60.7	72.3	83.4

Table 6: Extraction results using only HTML5 `article` tags.

This further demonstrates that the nature of the Web is changing, and as a result, our thinking about content extraction must change too.

Cohort	Extractor	Year	Lustrum 2000 Prec	2000 Rec	2000 Acc	2005 Prec	2005 Rec	2005 Acc	2010 Prec	2010 Rec	2010 Acc	2015 Prec	2015 Rec	2015 Acc
	All Text	–	45.65	100	45.65	38.33	100	38.33	25.78	100	25.78	20.14	100	20.14
2000	BTE	2001	76.36	92.74	82.74	58.15	89.37	72.22	34.32	88.67	53.92	20.67	85.47	44.05
2000	LSI	2001	83.37	89.79	87.87	64.19	88.71	77.32	43.35	80.47	65.7	23.66	77.51	48.56
2000	DSC	2002	85.42	83.25	86.2	66.88	82.98	77.55	46.37	75.74	71.09	23.89	72.97	50.15
2005	KFE	2005	74.21	75.88	83.34	50	69.95	70.71	35.28	63.45	65.4	19.78	64.92	47.72
2005	LQF	2005	71.11	93.81	81.49	56.57	92.39	72.23	38.68	85.44	61.94	20.91	84.47	45.24
2005	ADSC	2007	74.39	92.91	83.68	57.68	91.36	73.51	36.95	86.74	59.28	20.27	85.59	44.25
2005	CCB	2008	85.28	86.95	88.26	65.79	84.91	77.78	44.45	77.06	68.95	22.92	74.43	48.72
2005	RR	2008	81.97	92.11	88.96	70.73	89.75	86.54	61.32	76.57	70.82	47.75	86.25	79.70
2010	CETR	2010	86.74	85.18	88.98	76.05	82.08	85.13	59.01	81.32	80.79	54.66	67.86	88.03
2010	CETR-1D	2010	85.3	85.62	88.55	76.23	82.64	85.41	59.31	80.35	80.89	56.57	67.13	87.78
2010	CETR-Th	2010	89.92	81.95	89.16	82.1	77.52	86.74	65.63	78.31	84.21	57.42	72.76	89.6
2010	BP	2010	93.51	85.92	92.26	91.84	82.64	92.45	79.12	75.72	88.86	83.17	68.84	93.74
2010	BP-AE	2010	94.76	87.11	92.86	92.97	84.25	92.54	94.99	69.39	91.35	85.79	63.21	92.96
2010	BP-Sen	2010	97.37	84.43	92.78	97.47	81.71	93.53	97.19	66.84	91.26	89.06	61.33	93.09
2010	BP-Canola	2010	93.43	87.36	92.58	88.09	84.56	90.97	77.33	77.47	88.71	68.74	71.47	92.02
2010	BP-15W	2010	94.5	83.7	91.55	89.09	80.62	90.22	82.1	74.04	89.37	73.89	68.51	92.4
2010	BP-W	2010	91.45	88.83	92.51	88.31	86.12	91.76	81.79	78.54	89.58	83.31	70.84	93.97
2015	eatiht	2015	81.89	76.3	80.17	82.04	80.04	83.76	93.75	69.18	91.13	88.48	62.93	94.39

Table 5: Precision, recall and accuracy breakdown by lustrum (*i.e.*, the 5-year period in which data was collected) and cohort (*i.e.*, the set of extractors that were developed in the same time period)

3.2 Discussion

The main critique of wrapper induction methods is that they frequently require re-training. In response many heuristic/feature engineering approaches have been developed that are said to not require training and simply work out of the box.

These results underscore a robustness problem in Web content extraction. Ideally, Web science research should be at least partially invariant to change. If published content extractors are to be adopted and widely used they ought to be able to withstand changing Web standards. Wrapper induction techniques admit this problem; however, we find that heuristic content extractors are prone to obsolescence as well.

4. CONCLUSIONS

We conclude by recapping our main findings.

First, we put into concrete terms the changes to the form and function of the Web. We argue that most content extraction methodologies, by their reliance on unrendered, downloaded HTML markup, do not count very large portion of the final rendered Web page. This is due to the Web's increasing reliance on external sources for content and data via JavaScript, iframes, and so on.

Second, we find that although wrapper induction techniques are prone to breakage and require frequent retraining, the heuristic/feature engineering extractors studied in this paper, which argued to not require training at all, are also quickly obsolete.

4.1 Recommendations for future work

We argue that the two findings presented in this paper be immediately addressed by the content extraction community, and we make the following recommendations.

1. Future content extraction methodologies should be performed on completely rendered Web pages, and should therefore be created as Web browser extensions or with a similar rendered-in-browser setup using a headless browser like PhantonJS, etc. This methodology will allow for all of the content to be loaded so that it may be fully extracted. A browser-based content extractor might operate similar to the popular AdBlock software, but only render content rather than simply removing blacklisted advertisers. Aside from executing JavaScript and gathering all of the external resources, a browser based content extractor would also allow for a visual-DOM representation that may improve extraction effectiveness.

2. Future content extraction studies should examine Web pages and Web sites from different time periods to measure the overall robustness of the dataset. This is a difficult task and is perhaps contrary to the first recommendation because the external data from old Web pages may not be be easily rendered because the external may sources cease to exist. Nevertheless, it is possible to denote Web pages which have not changed via through Change Detection and Notification (CDN) systems [4] or through Last-Modified or ETag HTTP headers.

3. With the adoption semantic tags in HTML5, such as `section`, `header`, `main`, etc., as well as the creation of semantic attributes within the `schema.org` framework, it is important to ask whether content extraction algorithms are still needed at all. Many Web sites have mobile versions that streamline content delivery and a large number of content provides have content syndication systems or APIs that deliver pure content. It may be more important in the near future to focus attention on structured data extraction from lists and tables [26; 15; 19; 3; 2] and integrating that data for meaningful analysis.

Content extraction research has been an important part of the history and development of the Web, but this area of study would greatly benefit by considering these recommendations as they would lead to new approaches that are more robust and reliable.

5. REFERENCES

[1] Z. Bar-Yossef and S. Rajagopalan. Template detection via data mining and its applications. In *WWW*, page 580, New York, New York, USA, May 2002. ACM Press.

[2] M. J. Cafarella, A. Halevy, and J. Madhavan. Structured data on the web. *Communications of the ACM*, 54(2):72–79, 2011.

[3] M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu, and Y. Zhang. Webtables: Exploring the power of tables on the web. *Proc. VLDB Endow.*, 1(1):538–549, Aug. 2008.

[4] S. Chakravarthy and S. C. H. Hara. Automating change detection and notification of web pages. In *17th International Conference on Database and Expert Systems Applications*, pages 465–469. IEEE, 2006.

[5] B. Chidlovskii, B. Roustant, and M. Brette. Documentum eci self-repairing wrappers: Performance analysis. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, SIGMOD '06, pages 708–717, New York, NY, USA, 2006. ACM.

[6] V. Crescenzi, G. Mecca, and P. Merialdo. RoadRunner: Towards Automatic Data Extraction from Large Web Sites. *VLDB*, pages 109–118, Sept. 2001.

[7] V. Crescenzi and P. Merialdo. Wrapper inference for ambiguous web pages. *Applied Artificial Intelligence*, 22(1&2):21–52, 2008.

[8] N. Dalvi, P. Bohannon, and F. Sha. Robust web extraction. In *SIGMOD*, page 335, New York, New York, USA, June 2009. ACM Press.

[9] H. Davulcu, G. Yang, M. Kifer, and I. V. Ramakrishnan. Computational aspects of resilient data extraction from semistructured sources (extended abstract). In *Proceedings of the Nineteenth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '00, pages 136–144, New York, NY, USA, 2000. ACM.

[10] S. Debnath, P. Mitra, and C. L. Giles. Automatic extraction of informative blocks from webpages. In *SAC*, page 1722, New York, New York, USA, Mar. 2005. ACM Press.

[11] A. Finn, N. Kushmerick, and B. Smyth. Fact or fiction: Content classification for digital libraries. In *DELOS Workshop: Personalisation and Recommender Systems in Digital Libraries*, 2001.

[12] D. Gibson, K. Punera, and A. Tomkins. The volume and evolution of web page templates. In *Special interest tracks and posters of the 14th international conference on World Wide Web - WWW '05*, page 830, New York, New York, USA, May 2005. ACM Press.

[13] T. Gottron. Combining content extraction heuristics. In *iiWAS*, page 591, New York, New York, USA, Nov. 2008. ACM Press.

[14] T. Gottron. Content Code Blurring: A New Approach to Content Extraction. In *DEXA TIR Workshop*, pages 29–33. IEEE, Sept. 2008.

[15] R. Gupta and S. Sarawagi. Answering table augmentation queries from unstructured lists on the web. *Proc. VLDB Endow.*, 2(1):289–300, Aug. 2009.

[16] W. Han, D. Buttler, and C. Pu. Wrapping web data into XML. *ACM SIGMOD Record*, 30(3):33, Sept. 2001.

[17] 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*, page 441, New York, New York, USA, Feb. 2010. ACM Press.

[18] N. Kushmerick. Learning to remove internet advertisements. In *Proceedings of the third annual conference on Autonomous Agents*, pages 175–181. ACM, 1999.

[19] G. Limaye, S. Sarawagi, and S. Chakrabarti. Annotating and searching web tables using entities, types and relationships. *Proc. VLDB Endow.*, 3(1-2):1338–1347, Sept. 2010.

[20] B. Liu, R. Grossman, and Y. Zhai. Mining data records in Web pages. In *SIGKDD*, page 601, New York, New York, USA, Aug. 2003. ACM Press.

[21] C. Mantratzis, M. Orgun, and S. Cassidy. Separating XHTML content from navigation clutter using DOM-structure block analysis. In *HT*, page 145, New York, New York, USA, Sept. 2005. ACM Press.

[22] Marco Baroni, F. Chantree, A. Kilgarriff, and S. Sharoff. CleanEval: a competition for cleaning webpages. In *International Language Resources and Evaluation*, 2008.

[23] L. Martin and T. Gottron. Readability and the Web. *Future Internet*, 4:238–252, 2012. Special Issue Selected Papers from ITA 11.

[24] R. Palacios. Eatiht. `http://rodricios.github.io/eatiht/`, 2015.

[25] A. G. Parameswaran, N. N. Dalvi, H. Garcia-Molina, and R. Rastogi. Optimal schemes for robust web extraction. *PVLDB*, 4(11):980–991, 2011.

[26] R. Pimplikar and S. Sarawagi. Answering table queries on the web using column keywords. *Proceedings of the VLDB Endowment*, 5(10):908–919, 2012.

[27] D. Pinto, M. Branstein, R. Coleman, W. B. Croft, M. King, W. Li, and X. Wei. QuASM: a system for question answering using semi-structured data. In *JCDL '02: Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries*, pages 46–55, New York, NY, USA, 2002. ACM Press.

[28] T. Weninger, W. H. Hsu, and J. Han. CETR. In *WWW*, page 971, New York, New York, USA, Apr. 2010. ACM Press.

[29] Y. Zhai and B. Liu. Web data extraction based on partial tree alignment. In *WWW*, page 76, New York, New York, USA, May 2005. ACM Press.