FOREST: Focused Object Retrieval by Exploiting Significant Tag Paths

Marilena Oita
Télécom ParisTech; CNRS LTCI
& CustomerMatrix
Paris, France
marilena.oita@gmail.com

Pierre Senellart
Télécom ParisTech; CNRS LTCI
& NUS; CNRS IPAL
Paris, France & Singapore
pierre.senellart@telecom-paristech.fr

ABSTRACT

Content-intensive websites, e.g., of blogs or news, present pages that contain Web articles automatically generated by content management systems. Identification and extraction of their main content is critical in many applications, such as indexing or classification. We present a novel unsupervised approach for the extraction of Web articles from dynamically-generated Web pages. Our system, called FOREST, combines structural and information-based features to target the main content generated by a Web source, and published in associated Web pages. We extensively evaluate FOREST with respect to various baselines and datasets, and report improved results over state-of-the art techniques in content extraction.

1. INTRODUCTION

Context. Textual Web content on modern Web sites is, in the overwhelming majority of cases, produced by dedicated content management systems (CMSs). Such software generates Web pages with news items, blog posts, wiki entries, forum messages, etc., by filling a template with information fetched from databases. This results in both structural and visual (i.e., presentation) similarities across generated Web pages. In this automatic Web page generation process, the original textual or structured content is turned into a full-fledged HTML document, where the content is hidden among the markup encoding of the site layout [17]. Such *boilerplate* content is meant to ensure a common layout of the site, or to add contextual information, navigation structure, and advertisements.

Structural similarities between pages generated within the same site [11] can easily be seen at the level of the DOM tree, the tree representation of the structure of an HTML document. Using structural similarities of pages, the "data region" delimiters can often be identified by tracing the variable content within a fixed template. Unfortunately, boilerplate changes from one page to another.

The challenge addressed in this paper is the identification of structural patterns within CMS-generated Web pages that locate informative content, rather than boilerplate.

WebDB'15, May 31, 2015, Melbourne, VIC, Australia

ACM 978-1-4503-3627-7/15/05 ...$15.00
DOI: http://dx.doi.org/10.1145/2767109.2767112

Proposal. In this paper we propose a fully automatic, unsupervised, technique to derive a wrapper for the main content of a Web page, exploiting similarities of pages of the same Web site or Web *source*. We refers to this main content as a *Web article*.

Typically, Web page segmentation and template removal approaches for content extraction are usually performed at single-page level, not taking advantage of the inherent structural patterns of pages that share the same template and present the same functionality. Additionally, these approaches tend to employ methods that heavily rely on quantitative analysis of text-related features and HTML heuristics, rather than qualitative features that may involve relevance analysis or cognitive measurements.

Our proposed method, called FOREST for *Focused Object Retrieval by Exploiting Significant Tag paths*, identifies the article object that is dynamically generated by a Web source, and expressed in associated pages. The location of the article is given by a wrapper that is induced for each source, based on an unlabeled sample of structurally-similar Web pages, that we further call sample pages. We aggregate the similarities of sample pages. As structural analysis on its own is not sufficient to identify the *main* content, we add a novel ranking scheme that is based on the amount of informative content that the instances of these patterns carry across sample input pages. An advantage that specifically applies to FOREST is that, once run on some sample pages containing Web articles, the result can generalize to any Web page sharing the same functionality and template, as long as the template of the source remains unchanged.

FOREST performs the following main steps per Web source: (i) automatically acquires relevant keywords for each of the sample pages; (ii) identifies from the sample tag trees significant structural patterns; (iii) ranks structural patterns based on a relevance measure based on information theory and statistics; (iv) infers a wrapper in the form of a generic XPath expression that characterizes the main content generation patterns of the source; (v) uses the wrapper to extract the main content from Web pages generated by that source.

Contributions. We outline the following contributions of this work: (i) a fully automatic method for identification of main content generation patterns; (ii) a novel measure of content relevance that assesses the informativeness of structural patterns; (iii) a new public dataset called RED, semi-automatically created, and centered around the notion of Web source, containing 1,000 Web annotated sample pages from 90 Web sites; (iv) extensive experiments showing the accuracy in terms of precision and recall, for our baselines and FOREST, on various datasets.

We next discuss the related work, before presenting FOREST in Section 3 and the evaluation experiments in Section 4. An extended presentation of this material (additional experiments and baselines, more detailed presentation of the algorithm) can be found in [28].

2. RELATED WORK

Automatic wrapper induction. Content extraction from Web pages has often been formalized as a wrapper induction problem [1, 22, 25]. In unsupervised settings, wrapper induction makes use of the common structure of various objects, either at single Web page level [10], or across different Web pages [12] to identify occurrence patterns [27]. Generally used for highly structured Web pages (e.g., the Deep Web, where pages are automatically generated by submitting a Web form), wrapper induction methods often use inter-object content changes as identification clues [1, 10]. These approaches do not extend well to Web pages containing textual articles. Indeed, they are typically less structured and, most importantly, incorporate various logical components: related stories, comments, tag cloud, etc., each having its own structural (therefore, change) patterns.

Web page segmentation. Template removal approaches filter common DOM subtrees from sample pages by cross-page clustering [7,35]. But besides the existence of static template-specific parts, a Web page presents multiple topically-disjunctive portions that may change from page to page. Segmentation algorithms, that operate at the level of a single page and perform the partition of a Web page into blocks ranked based on their importance, have considered the extraction of "informative blocks" [33], pagelets [2, 7], fragments [32] or articles [20,31] from Web pages. This identification is dependent on a palette of features: HTML heuristics [19, 30], visual clues [6,26,37], or popular "shallow" text features (e.g., link density or text length). Content features may be fed to a machine learning algorithm, for either clustering [3,4,36], or classification [5,33].

Other works [31] aim at extracting the Web article of a Web page by relying on the text density in subsequent Web page segments. Similarly, BOILERPIPE [20] employs densitometric features (e.g., average word length, absolute number of words, etc.) for the identification of rules that can classify text into content or boilerplate. Belonging to a more recent type of techniques, CETR [36] computes, per line of HTML code of the Web page, a tag ratio array. The resulting matrix is fed to a histogram clustering algorithm to identify the positions of extraneous (i.e., boilerplate) data. We add the publicly available BOILERPIPE and CETR state-of-the-art techniques to our baselines to compare to in Section 4.

Keywords and Web feeds. The use of terms that appear with a statistically unusual frequency in a text is a common practice in information retrieval. Using keywords to rank Web blocks that contain interesting content has been proposed in [30]. In contrast with techniques considering all tag paths as equally important [27], the use of keywords in defining *informative* structural patterns reduces the exploration space [8]. The query terms occurring in search logs have been proposed as a keywords in [8], with the aim of performing a structural clustering of the Web pages resulting from queries to a search engine. However, the access to search logs is limited to Web sites owners, or to search engines themselves. Keyword search over XML documents [18, 34] assume the keywords given in the input of the algorithm. The associated techniques are either focused on finding the smallest lowest common ancestor (SLCA) that contains all searched keywords [34], or are based on variations of PageRank [18].

The use of Web feeds for main content extraction has been proposed in [30], where linguistic clues and DOM heuristics have been employed to identify the block where the main content resides. Despite their potential, Web feeds have just recently achieved more attention in the content extraction task [9, 16]. First, due to the intrinsic metadata about the main content that they provide, and second, due to the type of Web pages that can be easily collected, and on which automated wrapper induction can be made.

Tools and standards. Some tools, such as the open-source Readability[1] or similar browser plugins, aim at presenting the main content of a Web page in a more readable way. These tools (at least those whose code is available) use a combination of heuristics, especially on tag names, site-specific parameters, and estimation of text or link density. Several recent development of Web technologies go in the direction of adding more semantics to the markup of a Web page to clearly identify the main content of a Web page. The HTML5 recommendation introduces the `<article>` tag to denote a Web page's or a section's main content. This is not, however, used consistently enough at the moment, to be of use for main content extraction at scale.

3. METHODOLOGY

In this section, we present the method used to obtain significant structural patterns from the tag-trees of sample pages, and the informativeness measure used to rank these patterns. The input to FOREST is a collection of n distinct sample HTML pages generated using the same template. We first use HTMLCleaner[2] to build a DOM tree out of HTML tag soup; resulting documents are further denoted d_j, $1 \leqslant j \leqslant n$.

3.1 Marker Keywords

We use keywords as markers of where the main content of a Web page may lie. As they can pinpoint potentially informative segments in a Web page, they provide us with a basis for defining a notion of content relevance at the level of DOM elements. In this article, we choose to exploit keywords that are automatically identified. We consider two different ways to do so: from a tf–idf analysis on sample pages, and from an external metadata-source like a Web feed (e.g., RSS or Atom) resource.

Our first method for acquiring keywords consists in applying a tf–idf analysis on sample pages. The aim is to obtain for each sample page its top-k weighted keywords. Towards this aim, after document tag tree cleaning, we apply tokenization, stop words removal, and stemming on the full text of each Web page. We index the resulting terms and rank them according to the tf–idf measure.

The second method supposes the existence of a Web feed (i.e., RSS or Atom) linked to our data source. In our context, that of CMS-enabled Web sites, e.g., news or blogs, this is very often the case. Web feed items summarize Web pages' "interesting" content through typical metadata; in particular, *title* (the title of the entry) and *description* (usually an excerpt of its text, automatically generated by the CMS) give us a direct hint about the main content in that page. In contrast with the tf–idf method mentioned earlier, without any global analysis on the textual content of sample pages, we process the *title* and *description* of an item page to obtain representative terms. In this process, possible HTML tags are stripped, we perform tokenization, and rank the resulting terms according to their *frequency*.

Whatever the origin of keyords, the top-k relevant terms for each page are called *markers* of the respective page. In the experiments, we fix the k threshold to 10 (but discuss other settings).

3.2 Structural Patterns Analysis

Our aim is to rank the various structural patterns of a publishing source according to their likelihood they lead to the main content, exploiting acquired keyword markers and the similarity across pages of the same source.

For each document, we extract all textual leaf nodes whose content matches at least one keyword. We call these nodes *significant*

[1] `http://lab.arc90.com/2009/03/02/readability/`
[2] `http://htmlcleaner.sourceforge.net/`

to make the distinction between the nodes worth exploring, and the rest. All ancestors of a significant node are also deemed significant. Indeed, if a textual leaf node is significant, that is, it contains one or more keywords, then the same holds for all its ancestors. The higher a node is in the tag tree hierarchy, the more keywords a node tends to contain; at the same time, its density in keywords decreases due to a larger amount of text.

Structural similarities across sample pages let us abstract out unessential differences in the HTML "encoding" of content in node elements. A DOM element is typically identified through a tag name (such as `div` or `li`) and a list of attributes (`id`, `style`, `class`, etc.). Their combination is however not always unique. Only the `id` attribute can be considered unique, while `class`, for instance, can be common to various elements in the tag tree. Paragraphs or table elements (e.g., `td`, `tr`), are rarely identified by a unique combination of attributes. We enforce the unicity of an element in a document by adding to each node, at document-processing time, a *dfs* and a *level* attribute. The *dfs* records the order of browsing of a DFS (depth-first search) walk starting from the root, and the level represents the depth of an element in the tag tree. This enforced structure is serialized into an XML document.

Contrarily to what one would expect, modern CMSs do not always generate Web pages with precisely the same HTML encoding. The attribute values generated for HTML elements may slightly vary from one sibling page to another. A simple example is that we could find a `<div>` elements with a *class* attribute value of "post-43" in one document, and "post-30" in another. We therefore abstract out these differences in attribute values by removing numbers.

We associate a *structural pattern* to each element in the tag tree, defined as an *XPath expression* constructed based on $\langle t, atts \rangle$, where t is the tag name and *atts* is a set of key-value pairs of attributes, as follows: (i) if the element has no other attributes than *dfs* then *atts* is set to $\{dfs = l\}$ where l is the *dfs* index of the current element; (ii) if the element has attributes, *atts* is set to the collection of key-value attribute pairs, excluding *dfs*. The structural pattern associated to the element is then: `//`t`[`att_1`]`$\ldots$`[`att_l`]` where $atts = \{att_1, \ldots, att_l\}$ (writing in the XPath expression a key-value pair (k, v) as k`="`v`"`).

3.3 Content Relevance Measures

We tackle the main content identification issue by seeing it as a ranking problem of structural patterns leading to significant elements. We introduce next two content relevance measures that estimate the importance of a structural pattern sp_i.

Statistical keywords density. We fix a significant node element e_i in a document d_j. Let x be the number of keywords in e_i's text, counted *with their multiplicity*. All other terms not identified as keywords represent non-significant terms, be y their number. Then $N = x + y$ is the total number of terms in the text of e_i. We analogously denote the number of significant and non-significnt terms in the *whole* document d_j in which e_i occurs, as X, and Y respectively.

One of the most natural ways to determine whether a node is interesting is to compute its density in keywords, i.e., $\frac{x}{N}$. However, when N is small, this measure is imprecise, due to lack of observations: a node formed of a single keyword has poor chances to be the most significant node of the document. In such context, we use *Jeffrey's add-half rule* [21] that represents a better statistical estimator for the proportion of significant terms in a node. Furthermore, when sampling N elements from a potentially larger set, we have a margin of error on this estimation, given by the standard deviation $\sqrt{\frac{f(1-f)}{N}}$ [15]. Within an interval of one standard deviation, we obtain a confidence of $\approx 70\%$ [15].

We therefore define our statistical keyword density J, as the worst-case estimator at 70% confidence of the density, including Jeffrey's add-half rule; if this value is less than 0, we fix it to 0:

$$J := \max\left(0, \frac{1}{N+1}\left(x + 1/2 - \sqrt{\frac{(x+1/2)\times(y+1/2)}{N}}\right)\right).$$

For example, if $x = 8$ and $y = 20$, $J(x,y)$ is comparable to that of a node with $x = 3$ keywords and $y = 5$ regular terms: the lower proportion ($\frac{2}{5}$ compared with $\frac{3}{5}$) is compensated by the larger number of observations. Even more interestingly, when $x = 1$ and $y = 0$ we have a $J \approx 0.32$, to be compared to the naïve density of 1: this element is indeed dense, but due to the low (zero here) number of non-significant terms, we cannot be sure of its importance.

Unexpectedness. Another approach to content relevance that we study is derived from the notion of *unexpectedness*. Coming from the cognitive model of *simplicity theory* [13] and information theory in general, this measure relies on the observation that humans tend to find a situation interesting when they perceive a discrepancy in complexity. That is, a situation is unexpected if it is simpler to describe than to generate. Assume a computation model given (say, Turing machine encodings for a given universal Turing machine). Given an object, we consider its generation complexity C_w (i.e., the size of the program that has generated it), and its description complexity C (i.e., its Kolmogorov complexity, the minimum size of a program that describes it); then the unexpectedness of this object is the difference between the two (note that we always have $C \leqslant C_w$).

We apply this to the simple setting of non-uniform binomial distributions, that corresponds to our context. The generation complexity corresponds (up to an additive constant) to the logarithm of the number of ways to draw $x+y$ elements out of a set of $X+Y$ elements: $C_w := \log(X+Y)^{x+y} = (x+y)\log(X+Y)$. The description complexity, on the other hand, represents the complexity of describing the content of the textual node, knowing that x terms are significant: it is the logarithm of the number of ways of choosing exactly x significant and y non-significant terms, that is: $C := x\log X + y\log Y$. The unexpectedness [13] is the difference between these two complexities: $U := (x+y)\log(X+Y) - x\log X - y\log Y$.

As a typical example, for a Web page with a total of 20 keywords and 100 regular terms, a node with 10 keywords and 26 regular terms will have an unexpectedness of 23 bits, which is definitely higher than a node with 3 keywords and 1 regular term of 6 bits.

Informativeness. In preliminary experiments, we have tested U and J separately, and combined into the global informativeness measure, to observe their performance. The common pattern is that J and U alone give less precise F_1-measure than their combination (see [28] for details on the experiments). Unexpectedness favors elements with a large amount of text content that is richer in signifiers than the typical distribution of signifiers on the Web page as a whole. This turns out to be complementary to the statistical density J, which generally favors nodes poor in non-significant terms. Therefore, the content-based informativeness of a structural pattern sp_i that represents $node_{ij}$ in a document d_j, is defined as: $I(sp_i, d_j) := J(sp_i, d_j) \times U(sp_i, d_j)$.

The informativeness of a structural pattern sp_i, $1 \leqslant i \leqslant m$ (where m is the total number of structural patterns that are shared by our sample pages), is then given by the sum of the products between the unexpectedness and the statistical keyword density of a document node that presents the structural pattern sp_i in document d_j. In terms of the number q_i of occurrences of a pattern sp_i across all documents, we set: $R(sp_i) := \sum_{j=0}^{n} I(sp_i, d_j) \times q_i \times level(sp_i)$. The role of the q_i factor is clear, since we want to give a bigger weight to structural patterns that are not only informative, but also very frequent. In addition, the *level* factor is a heuristic favoring nodes that are deeper in the DOM tree. The primary reason for this addition is that elements that are too high in the hierarchy (e.g, `<body>`) are

more unlikely to effectively identify the target article object because they are not discriminative. A similar decay factor has also been introduced in other works [18, 23], varying from a constant to a probability-like value.

Coverage of high-weighted keywords. Intuitively, the node which contains the main content of an article would be the smallest, lowest common ancestor [34] that has a maximal coverage of the keywords. Recall that we have computed with tf–idf the top-k weighted keyword representative for each sample document d_j. The *coverage* of a node selected by a pattern sp_i can be then defined as the sum of tf–idf weights $w(u)$ of the keywords u occurring in its textual content, normalized by the total number of keywords occurring in the text of that node. Aggregating the coverage of nodes selected by sp_i across all documents gives the following:

$$\text{Cov}(sp_i) = \sum_{j=1}^{n} \frac{\sum_{p=0}^{\text{nbKeywords}(sp_i(d_j))} w(keyword_p)}{\text{totalNbOfKeywords}(d_j)}$$

We can therefore rank the structural patterns with this coverage relevance measure in a new method that we dub FOREST$_{\text{Cov}}$, and include it as a baseline. The objective is to assess whether taking into account the weights of keywords is more beneficial than using their simple presence in the informativeness measure.

In a final step, the best ranked pattern (by informativeness or coverage) is selected. In evaluation settings, the corresponding XPath expression is applied on the sample documents generated by a Web source in order to extract their main content.

4. EXPERIMENTS

We next evaluate the FOREST system that implements the proposed technique. We include multiple technique variations that are meant to show the motivation for the algorithmic choices made in FOREST, and their possible use in various scenarios. With the online availability of state-of-the-art content extraction techniques such as CETR [36] and BOILERPIPE [20], we are given the opportunity to directly compare to these methods. We also include a basic DESCRIPTION baseline.

4.1 Methods and Baselines

FOREST. Our system has been implemented in Java. We test FOREST using two sources of keywords: extracted from the Web pages themselves using tf–idf, or from Web feed items metadata. For the case in which we have automatically identified some representative terms for each of the sample pages using tf–idf, we propose two measures of content relevance, yielding FOREST$_{\text{info}}$ based on informativeness with respect to markers of occurrence, and FOREST$_{\text{Cov}}$ based on weighted coverage. In the case of keywords extracted from Web feeds, we name the method FOREST$_{\text{feed}}$.

ABSELEMS *and* ABSPATHS. These two baselines are meant to give an insight into the algorithmic choices that will finally conduct to best extraction efficiency. They follow the same logic of decomposing significant paths and rank structural patterns from the content point of view, with some differences in the way the structural patterns are chosen. ABSPATHS takes as structural pattern the full XPath expression associated with a root-to-leaf DOM path leading to a significant node. Therefore, the structural patterns are associated with tag tree paths rather than with elements. ABSELEMS, on the other hand, considers for the selection of structural patterns only significant leaf nodes. The goal behind this choice is to verify if it is necessary to associate structural patterns to all ancestor nodes, or it would be sufficient to do so only for the significant leaf nodes. For both ABSPATHS and ABSELEMS baselines, we have as many structural patterns as significant leaf nodes in the page. In contrast, FOREST$_{\text{info}}$ and FOREST$_{\text{Cov}}$ gather structural patterns from all ancestor nodes of a significant leaf node.

DESCRIPTION. In the RSS specification[3], each item has three compulsory elements: title, link, and description. While the *title* of an item gives the name of the Web article as it appears on the referenced Web page (pointed to by the *link* URL), the *description* often represents the first lines of the content of interest in that page. Sometimes, though, the description may contain the whole article content, often encoded in HTML. It becomes therefore important to verify if this is a prevalent situation. The DESCRIPTION baseline consists in just returning a Web feed title and description, to evaluate the cases when this information is enough for Web article extraction.

CETR. CETR [36] extracts main content based on the idea that a Web article is usually less segmentated than other regions, leveraging therefore the observation that consequently, the article would contain more text than HTML tags. For the comparison with the CETR clustering-based algorithm, we use the public implementation and dataset available online[4].

BOILERPIPE. [20] uses quantitative linguistics to identify fragmented, short text in blocks of a document as boilerplate, and remove it to obtain the article content of a Web page. In theory, BOILERPIPE extractors need to be trained for different types of content templates, in order to obtain adapted rules. However, the publicly available implementation[5] contains extractors that have all been trained for blogs and news Web pages, the precise context in which FOREST and the other baselines operate. Preliminary results showed that the *ArticleExtractor* gives the best results for our extraction task, thus it is selected to represent BOILERPIPE in the evaluation process.

4.2 Performance Metrics

The goal of FOREST is to select the best wrapper that describes the location of the main content in pages generated by a Web source. Specifically, FOREST-like methods output a wrapper (i.e, an XPath query) that targets one or various XML subtree(s) in the document. On the other hand, some of our baselines, like BOILERPIPE and DESCRIPTION, extract text directly. To normalize the outputs for all considered methods, we evaluate the wrappers learned by CETR and FOREST-like to produce text.

Methods are evaluated with respect to the amount of keywords shared with the gold standard (as in the analysis of CETR and BOILERPIPE). Having as reference the set of normalized tokens of the gold standard, and the textual output of each method to be compared, we compute the classic *precision* and *recall* of the main text extraction: $\text{Precision}(G,S) = \frac{|G \cap S|}{|S|}, \text{Recall}(G,S) = \frac{|G \cap S|}{|G|}$. Precision and recall are further summarized by their harmonic mean, the F_1 measure. Note that the precision we compute is exactly the ROUGE-N [24] measure used when comparing the performance of text summarization techniques.

4.3 Datasets

Most state-of-the-art techniques for article extraction operate at single Web page level. As a consequence, public datasets such as CleanEval[6] provide benchmarking support at single page level. FOREST however needs various sample pages generated by a Web source. We describe next the two datasets that we use.

[3] `http://cyber.law.harvard.edu/rss/rss.html`
[4] `http://www.cse.nd.edu/~tweninge/cetr/`
[5] `http://code.google.com/p/Boilerpipe/`
[6] `http://liste.sslmit.unibo.it/pipermail/sigwac/2011-May/000093.html`

CYAN. We include in the evaluation process the public dataset curated by [36], but initially provided by [31]. This dataset has been used in various related work techniques as a reference. It includes CleanEval-annotated pages, but also provides Web pages grouped in collections from various news Web sites. The CleanEval subset does not provide different Web pages for the same website, but rather disparate Web pages that are annotated, without a particular reference to their source. Luckily, besided CleanEval, the authors include 9 news and blog websites (bbc, myriad40, reuters, tribune, techweb, suntimes, nytimes, nypost, freep) that can be considered as sources in our context because they are grouping annotated pages.

We name the set of sources provided by [36] and directly downloaded from the CETR Web site[7] CYAN, for "CETR: Yet Another News" dataset. We randomly sample from CYAN sources a number of Web pages (typically, 10) that are structurally similar, property verified in a pre-processing phase using a structural fingerprinting.

RED. The public CYAN dataset provided by [36] contains only 9 news Web sources, so we created RED, for "RSS-based Experimental Dataset". For RED, we also annotate sample pages per source, and provide 91 blogs and news Web sites for the evaluation process. This dataset is publicly available for further usage[8].

RED is constructed using Web feeds, that are abundant in our news and blogs context. Feeds of Web sites are acquired in an semi-automatic manner by scraping the results of a feed meta-search engine like *Search4RSS*[9], in response to a keyword-based query. Another source of selection was technorati[10], where top-ranked blogs and news RSS were selected to get a coverage of popular sites. We have thus accumulated a total of 91 Web sources, and over 1,000 sample Web pages. RED is, to the best of our knowledge, the first feed-based dataset for Web article content extraction. Note that, although we use feeds to collect structurally-similar Web pages presenting Web articles, the presence of Web feeds is not a condition for the use of FOREST$_{\text{info}}$ or FOREST$_{\text{Cov}}$.

To verify the effectiveness of each tested methods on the RED dataset, we manually created a gold standard annotation for each of the sample pages. Web feeds have different number of items produced by their channel, therefore we have annotated between 2 and 20 Web pages depending on the feed's number of exposed items (knowing that the typical number of items in a feed is 10 [30]). An annotation of at least the half of the typical number of items belonging to the Web feed of a source is useful in the analysis of the number of pages that are necessary to reach a top extraction efficiency. The annotation is a particularly time-consuming process, since more than 1,000 Web pages have been manually annotated.

After a round of quality assurance to check that the guidelines were well understood, the annotation task is intuitive enough to reach a high-level of inter-annotator agreement: the precision from one annotator to another was 97%.

4.4 Experimental Results

For each Web source, results are given for the setting of a *maximum* 10 keywords and sample pages, unless otherwise specified.

Results on CYAN. As the output of all baselines has been normalized to plain text, we do the necessary parsing to extract the textual content of CYAN's gold standard. For that, we strip the HTML tags from the annotation, and apply typical normalization in both the gold standard and the extracted text to be compared with.

Table 1 presents the aggregated results for all relevant techniques included in the evaluation. We see that the winner on the CYAN dataset is BOILERPIPE. Indeed, its classifiers have been trained on news Web pages, the exact type of resource that we test here. Regarding CETR, we were unfortunately unable to reproduce th same results as in [36], on the same news sources from their dataset. One cause might be the fact that our gold standard considers only the text stripped out of HTML tags. Detailed verifications on the difference have also shown that, for the same input Web page as for the other techniques, CETR tends to throw more parsing exceptions. CETR gives however quite robust results, even though not as precise as those of BOILERPIPE.

Table 1: Mean precision, recall (%)

	CYAN		RED	
	Prec.	Rec.	Prec.	Rec.
ABSELEMS	65	68	87	93
ABSPATHS	58	64	72	74
FOREST$_{\text{info}}$	87	**99**	**88**	**98**
FOREST$_{\text{feed}}$			86	**98**
FOREST$_{\text{Cov}}$	78	89	**88**	**98**
BOILERPIPE	**94**	**97**	**89**	90
CETR	65	95	67	93
DESCRIPTION			92	31

FOREST$_{\text{info}}$ has performance almost comparable to that of BOILERPIPE, and better than CETR. Indeed, recall is even higher than for BOILERPIPE, at the expense of a significantly lower recall. FOREST$_{\text{Cov}}$ achieves relatively good performance, but lower than that of FOREST$_{\text{info}}$. ABSPATHS and ABSELEMS have widely varying performance, resulting in a low average precision and recall on this dataset. CYAN does not provide the necessary test conditions for DESCRIPTION and FOREST$_{\text{feed}}$.

Results on RED. We show in Table 1 the mean precision and recall on RED. We note that, since we have a dataset consisting of 90 independent Web souces and values of the order of 90%, the confidence interval at 95% probability (1.96 standard deviation) [15] is ± 0.06. To better understand the statistical significance of the results on RED, we show in Figure 1 a box plot of the F_1 measure of FOREST and its baselines, investigating the precise shape of the result distribution. On this graph, we show the 9th and 91th percentile (whiskers), 1st and 3rd quantile (box), as well as the median of the distribution of F_1 for each method.

Globally, we observe that FOREST$_{\text{info}}$ and FOREST$_{\text{Cov}}$ outperform the baselines, with a global F_1 measure of 92%. The similar performance of FOREST$_{\text{info}}$ and FOREST$_{\text{Cov}}$ suggests that, whatever their source, keywords can be put to use in the task of content extraction: whether weights are used, like in FOREST$_{\text{Cov}}$, or not, like in FOREST$_{\text{info}}$. FOREST$_{\text{feed}}$ manage a slightly lesser F_1 than the previous variants of FOREST, although only the source of keywords is different. Performing a tf–idf analysis seems therefore to be more useful than having the keywords given through the items metadata. FOREST$_{\text{feed}}$ still achieves a higher F_1 measure than BOILERPIPE, but functions in the same time in a more restrictive context, since it assumes sample pages are linked to a Web feed in order to have the metadata available. We note that all FOREST variants, excepting FOREST$_{\text{feed}}$, have the advantage of being independent of the Web feeds presence.

BOILERPIPE achieves a lower score than FOREST variants, despite the fact that the Web pages of our dataset (e.g., blog posts, news articles) match the kind of pages the *ArticleExtractor* is trained to work on. The explication that we find is related to the fact that the RED dataset contains not only news articles and blog posts, but

[7] `http://www.cse.nd.edu/~tweninge/cetr/`
[8] `http://dbweb.enst.fr/software/`
[9] `http://www.search4rss.com/`
[10] `http://technorati.com/blogs/top100/`

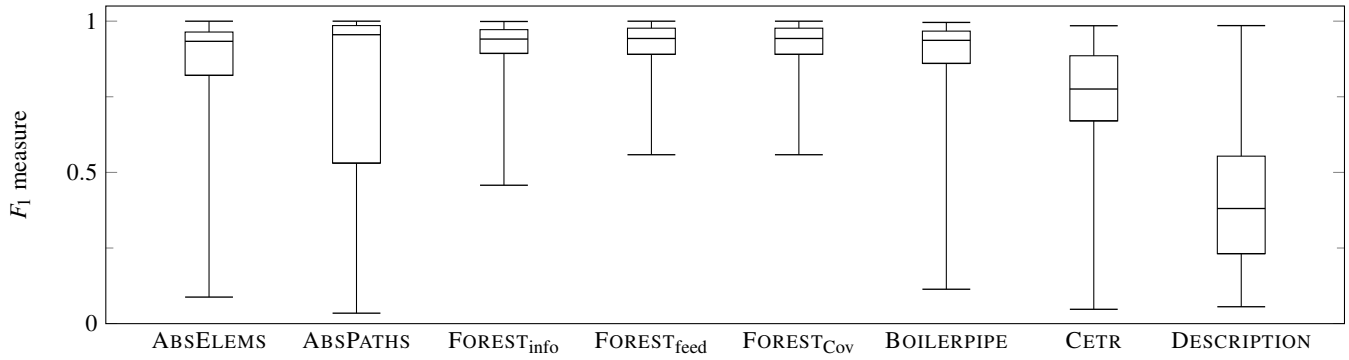

Figure 1: Box chart of F_1 measure on RED: 9th and 91th percentile (whiskers), 1st and 3rd quartile (box), median (horizontal rule)

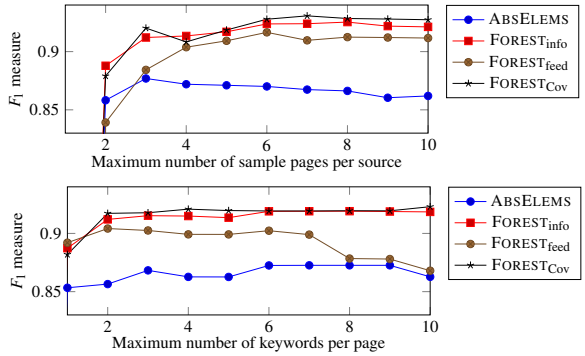

Figure 2: Evolution of the F_1 measure in function of the (maximum) number of sample pages (above) and keywords (below)

also and any other types of articles that were connected to a source through a Web feed. This results on a more template-generic dataset than the one on which BOILERPIPE was originally evaluated.

Another interesting feature shown in Figure 1 is related to ABSELEMS and ABSPATHS. ABSELEMS and ABSPATHS give high median and F_1 measures, but are not as robust as the previous variants of FOREST. Compared with FOREST$_{info}$ for instance, these results suggest the usefulness of considering in the acquisition of structural patterns all tag tree elements that are members of significant terminal paths.

CETR applied on RED gives most F_1 scores between 0.63 and 0.88, with a median F_1 of 0.76, which is somewhat poor.

As can be observed, DESCRIPTION performs very poorly, with a F_1 score greater than 50% on less than 25% of the corpus; its low precision suggests that a feed item description does not contain all text annotated as main content. Also, judging by the abysmal recall, we conclude that feed item descriptions are most often incomplete versions of the main content in a Web page. It is generally known [14] that for practical (e.g., the article can be very long) or commercial purposes (i.e., to attract site visitors), the majority of feed generators cut the item description to a couple of lines, representing the beginning of the main content. Indeed, a typical item description has a dedicated link to the unabridged version of the content that is being "advertised" through the feed.

Influence of the number of pages and keywords. To understand the impact of the number of sample pages and keywords on the effectiveness of FOREST, we plot in Figure 2 the F_1 measure for our baselines, depending on these parameters. Note that this only makes sense for FOREST, ABSELEMS, and ABSPATHS. Since the performance of ABSPATHS is quite low, we omit it from the figures.

Observe that as soon as there are at least two sample pages in the input (or three for FOREST$_{feed}$), FOREST reaches an F_1 score that is already above that of BOILERPIPE. Therefore, despite the fact that FOREST requires various sample pages, their number is rather small, which allows an easier and faster acquisition. The overall gain in using pattern reinforcement in ranking through a reduced number of sample pages is important since it avoids the need for specific training (as is the case for BOILERPIPE).

It is worth noting that FOREST requires at least two sample pages as input: this is helpful for the acquisition of *discriminative* keywords (e.g., for the tf-idf weightening), and, in general, allows the exploitation of the common template (it makes the structural patterns more conducive). FOREST efficiency keeps improving as the number of Web pages increases up to 6–8 pages, to reach a plateau around 8–10 pages. We also observe in Figure 2 (below) the impact that an incremental number of keywords per page has on the efficiency of our baselines. We note that as long as the number of keywords exceeds 5, the quality of the extraction is not affected. We also note that FOREST$_{feed}$ tends to have worse performance if the number of keywords is increased beyond 7-8, indicating that there are only so many relevant keywords in feed descriptions.

Scalability. Experiments on time efficiency (see [28]) show that BOILERPIPE takes the lowest time, followed by CETR, and then by FOREST variants. FOREST takes roughly four times more than BOILERPIPE in average on the sample pages. However, the advantage of FOREST is scalability. FOREST output characterizes the main content-related generation patterns of the source through an article wrapper. Once this wrapper has been obtained, content extraction will be reduced to a single operation, that of applying an XPath query on a Web document. This will eventually save time at each content request from Web pages pertaining to that source.

5. CONCLUSION

We presented in this paper FOREST, a robust and efficient unsupervised technique for the extraction of the main content from dynamically-generated Web pages. Our algorithm mingles wrapper induction with content analysis, for a sample of structurally-similar Web pages that are source-specific. We have successfully applied an adaptation of FOREST for the extraction of data records from deep Web [29]. For that, we used as *sample pages* the Web pages obtained in response to the submission of a Web form (e.g., Amazon advanced search form for books), and as *keywords* the terms employed during form submission. This demonstrates the flexibility of our approach, since sample pages can be easily obtained, and keywords can come from a large variety of contexts.

Acknowledgment

This work has been partly funded by the Télécom ParisTech Research Chair on Big Data and Market Insights. We are grateful to Internet Memory for their help in the annotation of the RED dataset.

6. REFERENCES

[1] A. Arasu and H. Garcia-Molina. Extracting structured data from Web pages. In *SIGMOD*, 2003.

[2] Z. Bar-Yossef and S. Rajagopalan. Template detection via data mining and its applications. In *WWW*, 2002.

[3] P. Bohunsky and W. Gatterbauer. Visual structure-based Web page clustering and retrieval. In *WWW*, 2010.

[4] E. Bruno, N. Faessel, H. Glotin, J. L. Maitre, and M. Scholl. Indexing and querying segmented web pages: the block Web model. *World Wide Web*, 14(5-6), 2011.

[5] D. Buttler, L. Liu, and C. Pu. A fully automated object extraction system for the World Wide Web. In *Distributed Computing Systems*, 2001.

[6] D. Cai, S. Yu, J.-R. Wen, and W.-Y. Ma. VIPS: a vision-based page segmentation algorithm. Technical Report MSR-TR-2003-79, Microsoft, 2003.

[7] J. Caverlee, L. Liu, and D. Buttler. Probe, cluster, and discover: focused extraction of QA-pagelets from the deep Web. In *ICDE*, 2004.

[8] D. Chakrabarti and R. R. Mehta. The paths more taken: Matching DOM trees to search logs for accurate Web page clustering. In *WWW*, 2010.

[9] R. Creo, V. Crescenzi, D. Qiu, and P. Merialdo. Minimizing the costs of the training data for learning Web wrappers. In *VLDS*, 2012.

[10] V. Crescenzi, G. Mecca, and P. Merialdo. RoadRunner: Towards automatic data extraction from large Web sites. In *VLDB*, 2001.

[11] V. Crescenzi, P. Merialdo, and P. Missier. Clustering Web pages based on their structure. *Data and Knowl. Eng.*, 54(3), 2005.

[12] D. de Castro Reis, P. B. Golgher, A. S. da Silva, and A. H. F. Laender. Automatic Web news extraction using tree edit distance. In *WWW*, 2004.

[13] A. Dimulescu and J.-L. Dessalles. Understanding narrative interest: Some evidence on the role of unexpectedness. In *CogSci*, 2009.

[14] E. Finkelstein. *Syndicating Web Sites with RSS Feeds for Dummies*. Wiley Publishing, Inc., 2005.

[15] D. Freedman, R. Pisani, and R. Purves. *Statistics*. W. W. Norton, 1998.

[16] G. Gkotsis, K. Stepanyan, A. I. Cristea, and M. S. Joy. Zero-cost labelling with web feeds for weblog data extraction. In *WWW*, 2013.

[17] S. Grumbach and G. Mecca. In search of the lost schema. In *ICDT*, 1999.

[18] L. Guo, F. Shao, C. Botev, and J. Shanmugasundaram. XRANK: ranked keyword search over XML documents. In *SIGMOD*, 2003.

[19] H.-Y. Kao, J.-M. Ho, and M.-S. Chen. WISDOM: Web intrapage informative structure mining based on document object model. *IEEE TKDE*, 17(5), 2005.

[20] C. Kohlschütter, P. Fankhauser, and W. Nejdl. Boilerplate detection using shallow text features. In *WSDM*, 2010.

[21] R. E. Krichevsky and V. K. Trofimov. The performance of universal encoding. *IEEE ToIT*, 27(2), 1981.

[22] N. Kushmerick, D. S. Weld, and R. Doorenbos. Wrapper induction for information extraction. In *IJCAI*, 1997.

[23] S.-J. Lim and Y.-K. Ng. An automated change-detection algorithm for HTML documents based on semantic hierarchies. In *ICDE*, 2001.

[24] C.-Y. Lin. ROUGE: a package for automatic evaluation of summaries. In *WAS*, 2004.

[25] B. Liu, R. L. Grossman, and Y. Zhai. Mining data records in Web pages. In *KDD*, 2003.

[26] R. R. Mehta, P. Mitra, and H. Karnick. Extracting semantic structure of Web documents using content and visual information. In *WWW*, 2005.

[27] G. Miao, J. Tatemura, W.-P. Hsiung, A. Sawires, and L. E. Moser. Extracting data records from the Web using tag path clustering. In *WWW*, 2009.

[28] M. Oita. *Deriving Semantic Objects from the Structured Web*. PhD thesis, Télécom ParisTech, 2012.

[29] M. Oita, A. Amarilli, and P. Senellart. Cross-fertilizing deep Web analysis and ontology enrichment. In *VLDS*, 2012.

[30] M. Oita and P. Senellart. Archiving data objects using Web feeds. In *IWAW*, 2010.

[31] J. Pasternack and D. Roth. Extracting article text from the Web with maximum subsequence segmentation. In *WWW*, 2009.

[32] L. Ramaswamy, A. Iyengar, L. Liu, and F. Douglis. Automatic detection of fragments in dynamically generated Web pages. In *WWW*, 2004.

[33] R. Song, H. Liu, J.-R. Wen, and W.-Y. Ma. Learning block importance models for Web pages. In *WWW*, 2004.

[34] C. Sun, C.-Y. Chan, and A. K. Goenka. Multiway SLCA-based keyword search in XML data. In *WWW*, 2007.

[35] K. Vieira, A. S. da Silva, and N. Pinto. A fast and robust method for Web page template detection and removal. In *CIKM*, 2006.

[36] T. Weninger, W. H. Hsu, and J. Han. Content extraction via tag ratios. In *WWW*, 2010.

[37] S. Yu, D. Cai, J.-R. Wen, and W.-Y. Ma. Improving pseudo-relevance feedback in Web information retrieval using Web page segmentation. In *WWW*, 2003.