Information Extraction from Webpages Based on DOM Distances*

Carlos Castillo[1], Héctor Valero[1], José Guadalupe Ramos[2], and Josep Silva[1]

[1] Universidad Politécnica de Valencia, Camino de Vera s/n, E-46022 Valencia, Spain
{carcasg1,hecvalli}@upv.es, jsilva@dsic.upv.es
[2] Instituto Tecnológico de La Piedad, La Piedad, México
guadalupe@dsic.upv.es

Abstract. Retrieving information from Internet is a difficult task as it is demonstrated by the lack of real-time tools able to extract information from webpages. The main cause is that most webpages in Internet are implemented using plain (X)HTML which is a language that lacks structured semantic information. For this reason much of the efforts in this area have been directed to the development of techniques for URLs extraction. This field has produced good results implemented by modern search engines. But, contrarily, extracting information from a single webpage has produced poor results or very limited tools. In this work we define a novel technique for information extraction from single webpages or collections of interconnected webpages. This technique is based on DOM distances to retrieve information. This allows the technique to work with any webpage and, thus, to retrieve information online. Our implementation and experiments demonstrate the usefulness of the technique.

1 Introduction

Information Extraction (IE) is one of the major areas of interest in both the web and the semantic web. The lack of real-time online applications able to automatically extract information from the web shows the difficulty of the problem. Current techniques for IE from Internet are mainly based on the recovering of webpages that are related to a specified query (see [7] for a survey). In this area, search engines such as Google or Bing implement very efficient and precise algorithms for the recovering of related webpages. However, for many purposes, the granularity level of the produced information is too big: a whole webpage.

In this work we try to reduce the granularity level of the information obtained. In particular we introduce a technique that given a collection of webpages, it extract from them all the information relevant for a given query and shows to the user in a new recomposed webpage.

* This work has been partially supported by the Spanish *Ministerio de Ciencia e Innovación* under grant TIN2008-06622-C03-02 and by the *Generalitat Valenciana* under grant PROMETEO/2011/052.

A. Gelbukh (Ed.): CICLing 2012, Part II, LNCS 7182, pp. 181–193, 2012.

In the semantic web setting, it is often possible to produce similar results composed of texts that answer a given question. However, these techniques often need to pre-process the webpages that are going to be queried. An ontological model is constructed and the knowledge is modeled and queried using languages such as RDF [8] or OWL [9]. This imposes important restrictions on the webpages that can be processed, and thus the implemented tools are usually offline tools. One reason is that most Internet pages have been implemented with plain (X)HTML. A similar problem is faced by the related techniques and tools that use microformats [10,11,12] to represent knowledge.

In this work we introduce a novel technique for IE that is based on DOM distances. Roughly speaking the technique looks for a term specified by the user, and it extracts from the initial webpage and some linked webpages those elements that are close to this term in the DOM trees of the webpages. Therefore, the technique relies on the idea that syntactically close means semantically related. This idea is also extended to distances between pages and domains using hyperlink distances. The main advantages of the technique are that it does not need to use proxies (as in [13]), it can work online (with any webpage) without any pre-compilation or pre-parsing phases (as in [14]); and that it can retrieve information at a very low level of granularity: a single word.

We are not aware of any tool that performs the kind of filtering that our system does. Other related approaches and tools [5] for web content filtering focus on the detection of one particular kind of content (such as porn or violence) in order to filter the whole webpage from a list of webpage results. Therefore, they do not decompose a webpage and filter a part of it as we do. Similar approaches are based on the use of neural networks [4] and application ontologies [6].

There are some works specialized for a particular content (such as tables) that are somehow related to our work. They do not focus on filtering but in content extraction from tables [1], or in wrappers induction [2,3]. In general, they detect a particular content in tables and extract it to be used as a database.

2 Information Extraction Based on DOM Distances

Our technique is based on the Document Object Model (DOM) [15] which is an API that provides programmers with a standard set of objects for the representation of HTML and XML documents. Our technique is based on the use of DOM as the model for representing webpages. For the sake of concreteness, in the following we will assume that a DOM tree is a data structure that represents each single element of a webpage with a node labelled with a text. This simplification assumes that all nodes have a single attribute, and it allows us to avoid in our formalization and algorithms low-level details such as the distinction between different kinds of HTML elements' attributes. For instance, in our implementation we have to distinguish and query different properties depending on the element that we are analyzing, e.g., image nodes are queried by using their *alt, longdesc* and *src* attributes.

Definition 1 (DOM Tree). *A DOM tree $t = (V, E)$ is a tree whose vertices V are nodes labelled with HTML elements connected by a set of edges E.*

We often refer to the label of a DOM node n with *l(n)*; and we refer to the root of a DOM tree t with *root(t)*. We also use the notation $n -^x n'$ to denote that there exists a path of size less or equal to x between nodes n and n'. If the path is of size x, then we say that the DOM distance between n and n' is x. Edges are represented as $(n \to n')$ with $n, n' \in V$. We use $\to^*$ to denote the reflexive and transitive closure of $\to$.

Definition 2 (Webpage). *A webpage is a pair (u, t) where u is a URL and t is a DOM tree.*

For simplicity, we assume that queries are composed of a single word. The extension of the technique for multiple words is trivial and it only requires the iteration of the method over the words of the query. This has been already done in our implementation, and thus, the interested reader is referred to its (open) source code for implementation details.

Definition 3 (Query). *A query is a pair (w, d) where w is a word that is associated with the information which is relevant for the user; and d is an integer that represents the tolerance required in the search.*

In our setting, the *tolerance* represents the maximum DOM distance allowed. The tolerance is used to decide what elements of the DOM tree are related to the user specified word. With tolerance=0, only elements that contain the specified word should be retrieved. With tolerance=1, only elements that contain the word and those that are in a distance of 1 to them should be retrieved, and so on.

Algorithm 1 implements our method for information extraction of single webpages. Clearly, this algorithm has a cost linear with the size of the DOM tree. In essence, it finds the *key_nodes* that are those whose label contains the searching word. From these nodes, the *relevant_nodes* are computed which are those whose DOM distance to the key nodes is equal or lower than the tolerance specified by the user. This idea is an important contribution of this technique because it is a novel method to retrieve semantically related information. Our experiments and implementation together with massive use of anonymous users demonstrate the practical utility of this DOM distance. All the *ancestors* and *successors* of the relevant nodes form the final nodes of the filtered DOM tree. The final edges are those induced by the final set of nodes. Therefore, the final webpage (that we will call in the following *slice*) is always a portion of the original webpage, and this portion keeps the original structure of information because the paths between retrieved elements are maintained.

In order to extend our algorithm for information extraction of interconnected webpages, in the following we will assume that the user has loaded a webpage (that we call *initial webpage*) and she specifies a query to extract information from this webpage, the webpages that are linked to it (either as incoming or outgoing links), the webpages included in it (e.g., as frames or iframes) and the webpages to which it belongs (e.g., as a frame or an iframe). We call all these pages the *interconnected webpages*; and observe that they are not necessarily in the same domain.

Algorithm 1. Information extraction from single webpages

Input: A webpage $P = (u, t)$ and a query $q = (w, d)$
Output: A webpage $P' = (\perp, t')$
Initialization: $t = (v, e), t' = (\emptyset, \emptyset)$

(1) $key\_nodes = \{n \in v \mid l(n) \text{ contains } w\}$
(2) $relevant\_nodes = \{n \in v \mid n -^d n' \wedge n' \in key\_nodes\}$
(3) $ancestors = \{n \in v \mid n_0 \rightarrow^* n \rightarrow^* n_1 \wedge n_0 = root(t) \wedge n_1 \in relevant\_nodes\}$
(4) $successors = \{n \in v \mid n_0 \rightarrow^* n \wedge n_0 \in relevant\_nodes\}$
(5) $edges = \{(n, n') \in e \mid n, n' \in (successors \cup ancestors)\}$

return $P' = (\perp, (successors \cup ancestors, edges))$

Frames and iframes can be modeled by considering that their DOM trees are subtrees of the webpage that contains them. Therefore, Algorithm 1 is able to extract relevant information from composite webpages structured with frames. For hyperlinks, we can assume that the label of some nodes in a DOM tree is a link pointing to a webpage. This is enough to define the notions of reachable webpages and search hyperspace used in our information extraction algorithm.

Definition 4 (Reachability). *Given a webpage P_0 we say that webpage P_n is reachable from P_0 if and only if* $\exists\ P_0, P_1, \ldots, P_n \mid \forall\ P_i = (u, (V, E)), 0 \leq i \leq n-1, \exists v \in V\ .\ l(v)$ *contains* $u' \wedge P_{i+1} = (u', t)$.

Roughly speaking, a webpage is reachable from another webpage if it is possible to follow a sequence of hyperlinks that connect both pages from the later to the former.

Definition 5 (Search Hyperspace). *Given a webpage $P = (u, t)$ the search hyperspace of P is the set of webpages that either are reachable following hyperlinks from nodes of P, or that can reach P from their hyperlinks.*

The search hyperspace is the collection of webpages that are related to the initial webpage, and that should (ideally) be inspected by our information extraction algorithm. However, the search hyperspace of a webpage is potentially infinite (specially when we surf dynamic webpages [16]), and it is often huge. Therefore we need to reduce it by discarding some of the hyperlinks. In addition, we want our technique to be online. This implies that time response is a critical factor, but the analysis of a webpage implies loading it, which is a time-consuming task. Therefore, reducing the number of webpages that are analyzed is a major objective of the technique.

With this aim, we define an hyperDOM distance between nodes of the search hyperspace. This distance is used to decide what hyperlink nodes are more related to the query specified by the user and should be explored. The others are discarded. Using syntax distances to approximate semantic relations is an idea that is supported by experimental evaluation of different works. For instance, Micarelli and Gasparetti [17] obtained empirical results demonstrating that webpages pointed by closer hyperlinks are more related semantically than webpages

pointed by hyperlinks that are syntactically separated. In order to define an hyperDOM distance, we use the following concepts:

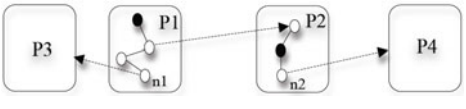
- ***DOM distance (dT):*** It is the length of the path between two nodes of a DOM tree.
- ***Page distance (dP):*** It is the lower number of hyperlinks that must be traversed to reach one webpage from another webpage.
- ***Domain distance (dD):*** It is the lower number of domains that must be traversed to reach one webpage from another webpage following a path of hyperlinks.

We use the initial webpage and the key nodes as the reference to compute distances. Therefore, for a given node, its DOM distance is the length of the path between this node and the closest key node in its DOM tree; and the page and domain distances are taken with respect to the initial webpage.

Definition 6 (HyperDOM Distance). *Given a DOM node n, the hyperDOM distance of n is* $D = d_T + K_P \cdot d_P + K_D \cdot d_D$ *where* K_P *and* K_D *are numeric constants used to weight the equation. The significance S of a DOM node is the inverse of its hyperDOM distance* $S = 1/D$.

Constants K_P and K_D determine the importance that we give to the fact that the word specified by the user is in another page, or in another domain.

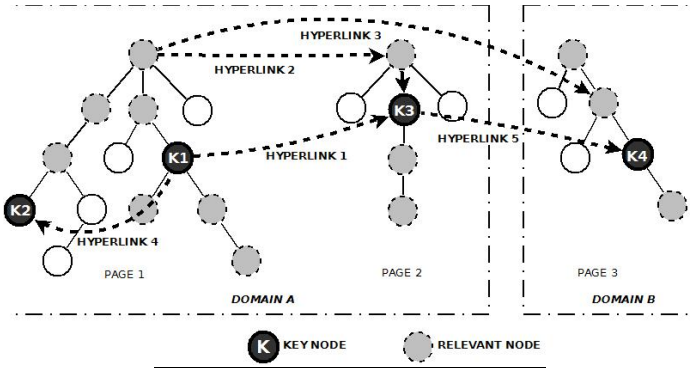
Example 1. Consider the following search hyperspace:



where two nodes contain the word specified by the user (those in black); the first node is in the initial webpage (P1), and the second node is in webpage P2 and thus it has a page distance of 1. Now, observe that nodes n1 and n2 are hyperlinks to other webpages. The question is: *which hyperlink is more related to the query of the user and should be explored first by the algorithm?* The answer is clear: the most relevant node and thus with a smaller hyperDOM distance. According to Definition 6, significance strongly depends on the values of the constants K_P and K_D. Assuming that all the webpages are in the same domain and if $K_P = 1$, then D(n1)=3 and D(n2)=2, thus n2 is more significant. In contrast, if $K_P = 10$, then D(n1)=3 and D(n2)=11, thus n1 is more significant.

After several experiments and intensive testing we took the following design decisions:

1 Those hyperlinks that are in the initial webpage are more important than those in another webpage. And the same happens as the page distance is increased. Hence, the DOM distance is more important than the page distance.



Hyperlink	d_T	d_P	d_D	D	S
1	0	0	0	0	∞
2	2	0	0	2	0.5
3	2	0	0	2	0.5
4	0	0	0	0	∞
5	0	1	0	10^6	$1\backslash 10^6$

Fig. 1. Relevant information hyperlinked through different pages and domains

2 Those hyperlinks that are in the same domain as the initial webpage are more important than those in another domain. And the same happens as the domain distance is increased. Hence, the page distance is more important than the domain distance.
3 The algorithm should never analyze a webpage with a page distance greater than 5. This is also supported by previous studies (see, e.g., Baeza and Castillo's discussion in [16]) that demonstrate that, in general, three to five levels of navigation are enough to get 90% of the information which is contextually related with the webpage selected by the user for the web search.

Therefore, considering the amount of nodes in a webpage, we take the following values: $K_P = 10^6$ and $K_D = 10^9$. The amount of DOM nodes in a webpage is usually lower than 10^3, thus, 10^6 ensures that the distance of two different pages is always greater than the distance of two nodes in the same webpage. Similarly, the amount of webpages analyzed in our method is usually lower than 10^2, thus, 10^9 ensures that the distance of two different pages analyzed in different domains is always greater than the distance of two different pages analyzed in the same domain. Hence, $D = d_T + 10^6 \cdot d_P + 10^9 \cdot d_D$

Example 2. Consider an initial webpage P1 and its search hyperspace shown in Fig. 1. Assume that Algorithm 1 has analyzed the three webpages and thus, dark nodes are relevant (key nodes are black) and white nodes are discarded. In order to determine what hyperlinks are more relevant, we compute the significance of their DOM nodes (see the table). This information is used to decide what hyperlinks must be analyzed first. Observe in the example that the hyperDOM

distance of node k4 is $0 + 1 * 10^6 + 1 * 10^9$. This node has a lower significance because it is in another domain. Note also that the significance of hyperlinks is computed from the source node (even though a hyperlink relates two DOM nodes, the HTML element that represents the hyperlink is in the source).

In a DOM tree we can distinguish between hyperlinks that belong to the slice and hyperlinks that do not belong to the slice. Those hyperlinks that do not belong to the slice are often related to webpages of none interest for the user. Therefore, to ensure the quality of the retrieved information we take a fourth design decision:

4 Hyperlinks that do not belong to the slice are discarded.

One important problem of extracting information from webpages happens in presence of dynamic webpages: A dynamic webpage could generate another dynamic webpage that contains the word specified by the user. This new dynamic webpage could do the same, and so on infinitely. This situation is known as black hole because robots searching in these webpages have an infinite search space where they always find what they are looking for. Therefore they are trapped forever if no limit is specified in the search [16]. Observe that the combination of design decisions 3 and 4 avoids this problem because the search is stopped when a webpage does not contain key nodes, or when its page distance is greater than 5. In addition, there is a fifth design decision related to the time response of the technique. Usability rules [18] establish that 10 seconds is (as an average) the time limit that users spend waiting for a webpage to be loaded. Therefore,

5 The maximum time spent to retrieve and show the information is 10 seconds.

The time used to show the retrieved information is constant, but the time used to load a webpage is variable. Therefore, the technique uses a mechanism to iteratively load webpages in significance order and extract information from them. When the time spent is close to the limit, the technique must stop the analysis.

Algorithm 2 summarizes the technique for information extraction of interconnected webpages. It uses the following functions that implement the ideas and equations explained in this section: *timeout()* controls that the algorithm never runs more than 10 seconds[1]. When the time is over, it returns *True*. *getSlice()* computes a slice of a webpage with Algorithm 1. *show()* shows in the browser a collection of DOM nodes. It should be implemented in a way that visualization is incremental. *getLinks()* extracts the link nodes of a set of nodes. *getMostRelevantLink()* computes the hyperDOM distance of a set of nodes to determine what is the most relevant node. *load()* loads a webpage.

[1] 10 seconds is the default time used in our implementation, but it can be set to any value (e.g., hours). In this case, constants K_P and K_D are redefined to ensure that pages in different domains are farther (with the hyperDOM distance) than pages in the same domain.

Algorithm 2. Information extraction from multiple webpages

Input: A set of interconnected webpages with an initial webpage P, and a query q
Output: A webpage P'
Initialization: $currentPage = P,\ pendingLinks = \emptyset$

while $not(timeout())$
(1) $relevantNodes = getSlice(currentPage, q)$
(2) $show(relevantNodes)$
(3) $pendingLinks = pendingLinks \cup getLinks(relevantNodes)$
(4) $link = getMostRelevantLink(pendingLinks)$
(5) $pendingLinks = pendingLinks/link$
(6) $currentPage = load(link)$

return P' (it is incrementally shown by the *show* function)

2.1 Visualization of the Relevant Information

Algorithm 2 is able to collect all the relevant DOM nodes of a set of webpages. Moreover, for each page, we know that the slice extracted is a valid webpage according to Algorithm 1. In addition, the information extracted is semantically related via hyperlinks and the semantic relation is weighted with the computed significance for each DOM node. Therefore, it is possible to use standard techniques for hierarchical visualization of the retrieved information. In our implementation reconstructing DOM trees is possible thanks to the DOM API's command:

documentNew.appendChildNode(documentOld.getElementById('myID'))

The command *documentOld.getElementById* allows us to extract from a DOM tree a specific element with a particular identifier *myID*. Then, the properties of this node can be queried, and if necessary, it can be inserted into another DOM tree with the command *documentNew.appendChildNode*. According to Algorithm 2, the visualization of the final webpage is done incrementally. For each analyzed webpage, we extract the slice with Algorithm 1, and then, this slice is inserted into the current webpage. Next, the webpage is refreshed, and thus, the technique produces results from the very beginning and, while the user inspects them, new results are added to the results webpage.

We have implemented two different algorithms to show the reconstructed webpage. The first one presents the information tabularly, the second one uses a hierarchical representation. Both algorithms retrieve information from different webpages and show it incrementally while it is being recovered. The main difference between them is the way in which the information is visualized in the browser.

Tabular Visualization. The lowest granularity level in this representation is a page. Basically, the final webpage is a linear succession of the filtered webpages. Each filtered webpage is considered as a whole, and thus, all the information that appeared together in the filtered webpage, is also together in the final webpage. The filtered webpages are ordered according to their navigational structure using a depth-first order.

Hierarchical Visualization. The lowest granularity level in this representation is a word. In this representation, the final webpage is a tree where the filtered webpages are organized. In contrast to the tabular representation, the filtered webpages can be mixed because each filtered webpage is placed next to the hyperlink that references it.

Example 3. Fig. 7 (left) shows a set of linked webpages where the dark part represents the relevant information, and its tabular representation of this relevant information. Fig, 7 (right) shows the hierarchical representation of the same set of webpages.

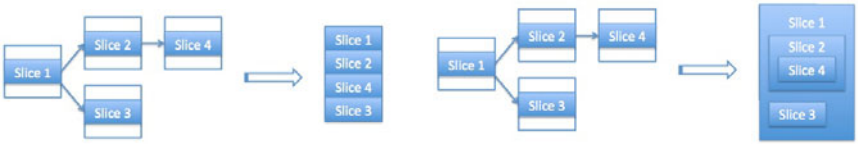


Fig. 2. Tabular visualization (left) and hierarchical visualization (right)

In Example 4 we show the complete process of information extraction.

Example 4. Consider again the initial webpage P1 and its search hyperspace of Fig. 1. Initially, Algorithm 2 extracts the slice of webpage P1. This slice is constructed from two key nodes (K1 and K2). Then, this information is shown to the user in a new webpage. Next, the algorithm tries to find the most relevant link to retrieve information from related webpages. In the table we see that the most relevant hyperlinks are H1 and H4. But H4 is discarded because it points to the initial webpage that has been already processed. Therefore, hyperlink 1 is selected and webpage P2 is loaded, processed and its slice shown to the user.

The information of webpage P2 is shown immediately after the information of K1, because, when this information is added to the webpage, it is placed close to the nodes that pointed to it. Hyperlink 2 is then discarded because it points to a webpage that has been already processed (P2). Because hyperlink 3 is more relevant than hyperlink 5, hyperlink 3 is selected first and webpage P3 is loaded, processed and shown to the user. Finally, hyperlink 5 is discarded because webpage P3 has been already processed. Hence, in the final webpage the slices are shown in order K1 K3 K4 K2.

Other models of visualization are possible and should be investigated. The presented models are designed to work in real-time because they work well when the amount of information shown is not too much (e.g., less than 20 slices). However, if the tool is used in batch mode (e.g., without time limitation), many webpages are filtered and the amount of information to be shown can be too much as to be shown in a single webpage; thus, it should be organized and probably indexed. For this, other models based on tiles [20] or clusters [19] would be more appropriate. Regarding the visualization of many slices, we are currently working

on a third visualization model called *site map*. Roughly, it produces an initial webpage with a site map with links that point to all the slices retrieved with the tool, and these slices are organized according to their original navigational map.

2.2 Implementation and Experiments

We have implemented the technique as an official plugin integrated in Firefox. The implementation allows the programmer to parameterize the technique in order to adjust the amount of information retrieved, the number of webpages explored, the visualization model and other functionalities. In order to determine the default configuration, it was initially tested with a collection of real webpages producing good results that allowed us to tune the parameters. Then, we conducted several experiments with real webpages. Concretely, we selected domains with different layouts and page structures in order to study the performance of the technique in different contexts (e.g., company's websites, news articles, forums, etc.).

For each domain, we performed two independent experiments. The first experiment provides a measure of the average performance of the technique regarding recall, precision and the F1 measure. The goal of this experiment was to identify the information in a given domain that is related to a particular query of the user. Firstly, for each domain, we determined the actual relevant content of each webpage by downloading it and manually selecting the relevant content (both text and multimedia objects). This task was performed by three different people without any help of the tool. The selected relevant content included all the information that each user thought was relevant for her. The DOM tree of the selected text was then built for each webpage. In a second stage, we used the tool to explore the webpages using Algorithm 2 and it extracted from them the relevant parts (according to the tool). Finally, we compared the DOM trees produced manually with those produced automatically by the tool. Table 1 summarizes the results obtained.

Table 1. Benchmark results

Domain	Query	Pages	Retrieved	Correct	Missing	Recall	Precision	F1
www.ieee.org	student	10	4615	4594	68	98.54 %	99.54 %	99.03 %
www.upv.es	student	19	8618	8616	232	97.37 %	99.97 %	98.65 %
www.un.org/en	Haiti	8	6344	6344	2191	74.32 %	100 %	85.26 %
www.esa.int	launch	14	4860	4860	417	92.09 %	100 %	95.88 %
www.nasa.gov	space	16	12043	12008	730	94.26 %	99.70 %	96.90 %
www.mityc.es	turismo	14	12521	12381	124	99 %	98.88 %	98.93 %
www.mozilla.org	firefox	7	6791	6791	14	99.79 %	100 %	99.89 %
www.edu.gva.es	universitat	28	10881	10856	995	91.60 %	99.79 %	95.51 %
www.unicef.es	Pakistán	9	5415	5415	260	95.41 %	100 %	97.65 %
www.ilo.org	projects	14	1269	1269	544	69.99 %	100 %	82.34 %
www.mec.es	beca	24	5527	5513	286	95.06 %	99.74 %	97.34 %
www.who.int	medicines	14	8605	8605	276	96.89 %	100 %	98.42 %
www.si.edu	asian	18	26301	26269	144	99.45 %	99.87 %	99.65 %
www.sigmaxi.org	scientist	8	26482	26359	241	99.08 %	99.54 %	99.30 %
www.scientificamerican.com	sun	7	5795	5737	97	98.33 %	98.99 %	98.65 %
ecir2011.dcu.ie	news	8	1659	1503	18	98.81 %	90.59 %	94.52 %
dsc.discovery.com	arctic	9	29097	29043	114	99.60 %	99.81 %	99.70 %
www.nationalgeographic.com	energy	12	41624	33830	428	98.75 %	81.27 %	89.16 %
physicsworld.com	nuclear	15	10249	10240	151	98.54 %	99.91 %	99.22 %

For each domain, the first column contains the URL of the initial webpage. Column `Pages` shows the number of pages explored by the tool in each experiment (the analysis time was limited to 10 seconds). Column `Query` shows the query used as the slicing criterion. Column `Retrieved` shows the number of DOM nodes retrieved by the tool; in the DOM model, the amount of words contained in a DOM node depends on the HTML source code of the webpage. It usually contains between one sentence and one paragraph. Column `Correct` shows the number of retrieved nodes that were relevant. Column `Missing` shows the number of relevant nodes not retrieved by the tool. Column `Recall` shows the number of relevant nodes retrieved divided by the total number of relevant nodes (in all the analyzed webpages of each domain). Column `Precision` shows the number of relevant nodes retrieved divided by the total number of retrieved nodes. Finally, column `F1` shows the F1 metric that is computed as $(2 * P * R)/(P + R)$ being P the precision and R the recall.

The first important conclusion of the experiments is that, in 10 seconds, the tool is able to analyze 13,3 pages as an average for each domain. Therefore, because the visualization algorithms are incremental, the first result is shown to the user in less than 1 second (10/13,3 seconds).

Results show that the tool produces a very high recall and precision. We were not surprised by the high precision of the tool because the syntactic matches with the DOM nodes ensures that the information retrieved is often very related to the user's query. But we were very excited with the recall being so high. Only in a few cases the recall was bellow 75%. The cause was the occurrence of synonyms that the tool is currently ignoring. Our next release will include a lexicon to solve this problem. In ten seconds results are very good because the tool explores webpages that are close to the initial webpage, and, in this search space, it is able to accurately detect semantic relations between pages.

After these good results, we were wondering whether this tool could be also used to retrieve information in a batch process (i.e., without a time limit, analyzing as many pages as possible). In this context, we wanted to know what is the page coverage of the tool. For this, we conducted a second experiment in which we retrieved information from the domains allowing the tool to explore as much as possible (i.e., restrictions 3 and 5 were ignored). Then, we collected the amount of webpages analyzed by the tool and compared it with the amount of (reachable) webpages in the whole domain. The later was computed with the Apache crawler Nutch [22]: the whole domain was indexed starting from the initial webpage and the amount of indexed documents was counted. The result was that the tool explored, as an average, 30% of the webpages in the search space of all the domains in Table 1. The cause is that the technique automatically discards many hyperlinks and concentrates on the most relevant search space; this is due to restriction 4, that prevents the tool to explore those webpages pointed by other webpages without relevant nodes. Relaxing restriction 4 would allow the tool to explore the whole search space, but precision would (probably) decrease significantly, because it would retrieve information from different contexts.

All the information related to the experiments, including the source code of the tool and other material can be found at: http://www.dsic.upv.es/~jsilva/web filtering

The official webpage of the tool at Firefox where the last stable release can be downloaded and where several comments and feedback from real users can be found at: https://addons.mozilla.org/en-US/firefox/addon/web-filtering-toolbar

3 Conclusions

This work introduces a novel information extraction technique based on syntax distances. The technique is able to work online and extract information from websites without any pre-compilation, labeling, or indexing of the webpages to be analyzed. Our experiments produced an F1 measure of 96%, demonstrating the usefulness of the technique. The analysis of the experimental results revealed that synonyms can cause a loss of recall. We are currently analyzing the impact of a lexicon. Using synonyms and semantic relations will allow us to increase the precision of our algorithms, but the efficiency of the technique will be affected. Empirical experimentation is needed to decide whether it is better to analyze many webpages without the use of a lexicon or few webpages with a lexicon. A balance between amount of information retrieved and the quality of this information must be studied. Our current implementation has been integrated in version 1.5 of the *Firefox WebFiltering Toolbar*. This tool is an official extension of the Firefox web browser that has been tested and approved by Firefox developers experts area, and that has more than 11.000 downloads at the time of writing these lines.

References

1. Dalvi, B., Cohen, W.W., Callan, J.: Websets: Extracting sets of entities from the web using unsupervised information extraction. Technical report, Carnegie Mellon School of computer Science (2011)
2. Kushmerick, N., Weld, D.S., Doorenbos, R.: Wrapper induction for information extraction. In: Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI 1997) (1997)
3. Cohen, W.W., Hurst, M., Jensen, L.S.: A flexible learning system for wrapping tables and lists in html documents. In: Proceedings of the international World Wide Web conference (WWW 2002), pp. 232–241 (2002)
4. Lee, P.Y., Hui, S.C., Fong, A.C.M.: Neural networks for web content filtering. IEEE Intelligent Systems 17(5), 48–57 (2002)
5. Anti-Porn Parental Controls Software. Porn Filtering (March 2010), `http://www.tueagles.com/anti-porn/`
6. Kang, B.-Y., Kim, H.-G.: Web page filtering for domain ontology with the context of concept. IEICE - Trans. Inf. Syst. E90, D859–D862 (2007)
7. Henzinger, M.: The Past, Present and Future of Web Information Retrieval. In: Proceedings of the 23th ACM Symposium on Principles of Database Systems (2004)

8. W3C Consortium. Resource Description Framework (RDF), www.w3.org/RDF
9. W3C Consortium. Web Ontology Language (OWL), www.w3.org/2004/OWL
10. Microformats.org. The Official Microformats Site (2009), http://microformats.org
11. Khare, R., Çelik, T.: Microformats: a Pragmatic Path to the Semantic Web. In: Proceedings of the 15h International Conference on World Wide Web, pp. 865–866 (2006)
12. Khare, R.: Microformats: The Next (Small) Thing on the Semantic Web? IEEE Internet Computing 10(1), 68–75 (2006)
13. Gupta, S., et al.: Automating Content Extraction of HTML Documents. World Wide Archive 8(2), 179–224 (2005)
14. Li, P., Liu, M., Lin, Y., Lai, Y.: Accelerating Web Content Filtering by the Early Decision Algorithm. IEICE Transactions on Information and Systems E91-D, 251–257 (2008)
15. W3C Consortium, Document Object Model (DOM), www.w3.org/DOM
16. Baeza-Yates, R., Castillo, C.: Crawling the Infinite Web: Five Levels Are Enough. In: Leonardi, S. (ed.) WAW 2004. LNCS, vol. 3243, pp. 156–167. Springer, Heidelberg (2004)
17. Micarelli, A., Gasparetti, F.: Adaptative Focused Crawling. In: The Adaptative Web, pp. 231–262 (2007)
18. Nielsen, J.: Designing Web Usability: The Practice of Simplicity. New Riders Publishing, Indianapolis (2010) ISBN 1-56205-810-X
19. Zhang, J.: Visualization for Information Retrieval. The Information Retrieval Series. Springer, Heidelberg (2007) ISBN 3-54075-1475
20. Hearst, M.A.: TileBars: Visualization of Term Distribution Information. In: Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems, Denver, CO, pp. 59–66 (May 1995)
21. Gottron, T.: Evaluating Content Extraction on HTML Documents. In: Proceedings of the 2nd International Conference on Internet Technologies and Applications, pp. 123–132 (2007)
22. Apache Foundation. The Apache crawler Nutch (2010), http://nutch.apache.org