Automatic Annotation of Content-Rich HTML Documents: Structural and Semantic Analysis

Saikat Mukherjee[1], Guizhen Yang[2], and I. V. Ramakrishnan[1]

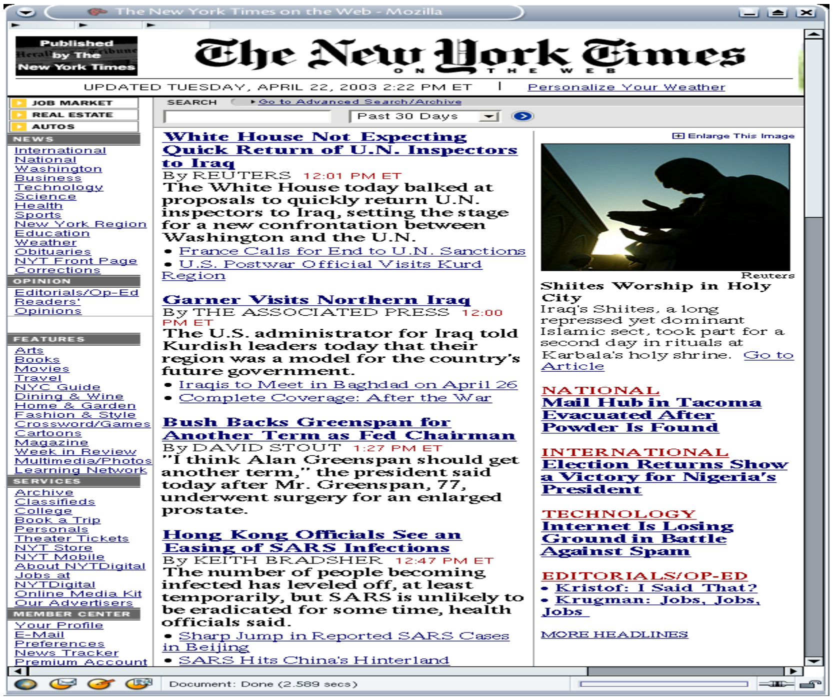
[1] Department of Computer Science
Stony Brook University, Stony Brook, NY 11794, U.S.A.
{saikat,ram}@CS.StonyBrook.EDU
[2] Department of Computer Science and Engineering
University at Buffalo, Buffalo, NY 14260, U.S.A.
gzyang@CSE.Buffalo.EDU

Abstract. Although RDF/XML has been widely recognized as the standard vehicle for representing semantic information on the Web, an enormous amount of semantic data is still being encoded in HTML documents that are designed primarily for human consumption and not directly amenable to machine processing. This paper seeks to bridge this semantic gap by addressing the fundamental problem of *automatically* annotating HTML documents with semantic labels. Exploiting a key observation that semantically related items exhibit consistency in presentation style as well as spatial locality in template-based content-rich HTML documents, we have developed a novel framework for automatically partitioning such documents into semantic structures. Our framework tightly couples structural analysis of documents with semantic analysis incorporating domain ontologies and lexical databases such as WordNet. We present experimental evidence of the effectiveness of our techniques on a large collection of HTML documents from various news portals.

1 Introduction

The Semantic Web espouses a vision [3] for next-generation information networks where content providers define and share machine processable data on the Web to empower a variety of automated tasks ranging from information integration to Web services. There have been significant activities in building the technological infrastructure for realizing the Semantic Web vision and applications based on it. Notable efforts include developing Semantic Web languages such as DAML+OIL [22, 23] based on Description Logic, and DAML Rules [16], which seeks to extend the Semantic Web with rule-based and object-oriented capabilities. Prototype systems, such as FaCT [21], $\mathcal{F}$LORA-2 [31, 30], and TRIPLE [28], are now available to drive the knowledge representation and reasoning component of the Semantic Web.

Semantic Web documents contain metadata to express the meaning of their content. Tools such as those pioneered by SHOE [20] and OntoBroker [13, 29] facilitate manual annotation of HTML documents with semantic markups. However, an enormous amount of semantic data (such as product descriptions and pricing information) is still being encoded in "plain" HTML documents. These documents are designed primarily for human consumption and hence, unlike annotated Semantic Web documents, are not machine understandable.



White House Not Expecting Quick Return of U.N. Inspectors to Iraq
By REUTERS 12:01 PM ET
The White House today balked at proposals to quickly return U.N. inspectors to Iraq, setting the stage for a new confrontation between Washington and the U.N.
- France Calls for End to U.N. Sanctions
- U.S. Postwar Official Visits Kurd Region

Garner Visits Northern Iraq
By THE ASSOCIATED PRESS 12:00 PM ET
The U.S. administrator for Iraq told Kurdish leaders today that their region was a model for the country's future government.
- Iraqis to Meet in Baghdad on April 26
- Complete Coverage: After the War

Bush Backs Greenspan for Another Term as Fed Chairman
By DAVID STOUT 1:27 PM ET
"I think Alan Greenspan should get another term," the president said today after Mr. Greenspan, 77, underwent surgery for an enlarged prostate.

Hong Kong Officials See an Easing of SARS Infections
By KEITH BRADSHER 12:47 PM ET
The number of people becoming infected has leveled off, at least temporarily, but SARS is unlikely to be eradicated for some time, health officials said.
- Sharp Jump in Reported SARS Cases in Beijing
- SARS Hits China's Hinterland

Shiites Worship in Holy City
Iraq's Shiites, a long repressed yet dominant Islamic sect, took part for a second day in rituals at Karbala's holy shrine. Go to Article

NATIONAL
Mail Hub in Tacoma Evacuated After Powder Is Found

INTERNATIONAL
Election Returns Show a Victory for Nigeria's President

TECHNOLOGY
Internet Is Losing Ground in Battle Against Spam

EDITORIALS/OP-ED
- Kristof: I Said That?
- Krugman: Jobs, Jobs, Jobs

MORE HEADLINES

Fig. 1. New York Times Front Page

In this paper we address the relatively unexplored problem of *automatically* annotating HTML documents, especially those that are machine generated from templates and contain *rich* semantic data. These kinds of documents are increasingly common nowadays since most content-rich Web sites (*e.g.*, news portals, product portals, etc.) are typically maintained using content management software that creates HTML documents by populating templates from backend databases. Figure 1, depicting a screen shot of the New York Times front page, is an example of a template-based content-rich HTML document. Observe that it has a news taxonomy (on the left in the figure) which does not change and a template for major headline news items. Each of these items begins with a hyperlink labeled with the news headline (*e.g.*, "White House ...") followed by the news source (*e.g.*, "By REUTERS ..."), followed by a timestamp and a text summary of the article (*e.g.*, "The White House today ...") and (optionally) a couple of pointers to related news.

Usually a content-rich HTML document (such as the one shown in Figure 1) comprises many semantic concepts (*e.g.*, "News Taxonomy" and "Major Head-

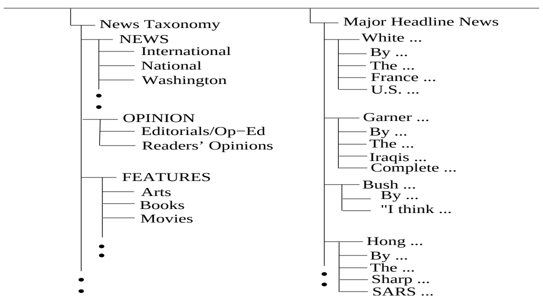


Fig. 2. A Fragment of the Semantic Partition Tree for New York Times Front Page

line News") and concept instances. These concepts and concept instances can be organized into a *semantic partition tree* (such as the one shown in Figure 2) which represents the "semantics" of a HTML document. In a semantic partition tree each partition (subtree) consists of items related to a semantic concept. For example, in Figure 2 all the major headline news items are grouped under the subtree labeled "Major Headline News". The problem addressed in this paper is to automatically transform template-based content-rich HTML documents into their corresponding semantic partition trees. Note that once a semantic partition tree is created, it is relatively straightforward to generate a corresponding new document (in RDF/XML or some other suitable formats) with semantic annotations.[3]

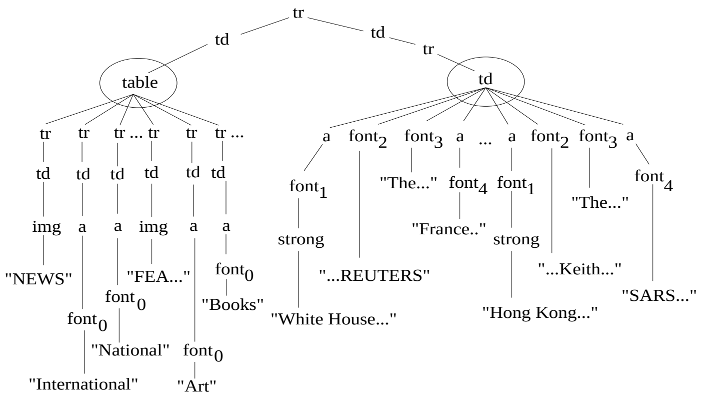


Fig. 3. DOM Tree Fragment of New York Times Front Page

There are two main tasks underlying the creation of a semantic partition tree from a HTML document: (i) identify segments of the document that correspond to semantic concepts; and (ii) assign labels to these segments. Informally, we

[3] However, the final document generation step is not the main focus of this paper.

say that several items are semantically related if they all belong to the same concept. Our solution to these problems is based on two important observations about semantically related items in template-based HTML documents, namely:

- *Semantically related items exhibit consistency in presentation style.* For example, observe the presentation styles of the items in the news taxonomy of the New York Times front page (on the left in Figure 1). The main taxonomic items, "NEWS", "OPINION", "FEATURES", ..., etc., are all presented in bold font. All the subtaxonomic items (*e.g.*, "International", "National", "Washington", ..., etc.) under a main taxonomic item (*e.g.*, "NEWS") are hyperlinks. A similar observation can also be made on all the major headline news items in the Figure 1.
- *Semantically related items exhibit spatial locality.* For example, when rendered in a browser (see Figure 1), all the taxonomic items are placed in close vicinity occupying the left portion of the page. Further observe that in the DOM tree (see Figure 3) corresponding to the HTML document, all these taxonomic items are grouped together under one *single* subtree rooted at the *table* node (shown in circle). Similarly, all the major headline news items are clustered under a different subtree rooted at the *td* node (shown in circle in Figure 3).

Our solution to automatically creating semantic partition trees exploits the two key observations above. The first observation leads to the idea of associating a type with every leaf node in the DOM tree. The type of a leaf node consists of the root-to-leaf path of this node in the DOM tree and captures the notion of consistency in presentation style. The second observation gives rise to the idea of propagating types bottom-up in the DOM tree and discovering structural recurrence patterns for semantically related items at the root of a subtree. The reader is referred to Section 2.1 for more details.

Summary of Our Approaches and Contributions

- Based on the idea of types and type propagation, we develop structural analysis techniques for automatically partitioning HTML documents into semantic structures. Our algorithm can also discover semantic labels and associate them with partitions when they are present in the document (*e.g.*, "NATIONAL", "INTERNATIONAL", ..., etc. appearing in the third column in Figure 1). (See Sections 2.1 and 2.2.)
- We develop semantic analysis techniques to factor in structural variations in concept instances (*e.g.*, the absence of pointers to related news in the third major headline news item in Figure 1 in contrast to the others). Our semantic analysis makes lexical associations via WordNet to more accurately put the pieces of a concept instance together. To assign informative labels that are not present in a HTML document (*e.g.*, "Major Headline News" in Figure 2) to partitions our semantic analysis makes concept associations by classifying the content of a partition using an ontology encoding domain knowledge. (See Section 2.2.)

– To accurately separate and aggregate instances of a semantic concept as well as uncover higher level concept instances in a complex ontology we propagate lexical and concept associations by coupling structural and semantic analysis to work in tandem. (See Section 2.2.)

The rest of this paper is organized as follows. Our framework for automatic semantic partitioning is presented in Section 2. In Section 2.1 we introduce structural analysis techniques based on sequential pattern discovery. In Section 2.2 we show how structural analysis can be coupled with semantic analysis to improve accuracy of partitioning. Experimental results appear in Section 3. Related work is discussed in Section 4. Finally, Section 5 concludes this paper.

2 A Framework for Automatic Semantic Partitioning

As we point out in Section 1, the principal step in automatic semantic annotation is to transform HTML documents into semantic partition trees. This is achieved in our framework through a combination of structural and semantic analysis techniques. First we illustrate the ideas underlying structural analysis.

2.1 Structural Analysis

Structural analysis is based on the key observation that in template-based content-rich HTML documents semantically related items exhibit consistency in presentation style and spatial locality. Such a property can be captured by a simple typing system for the nodes in the DOM tree (and its corresponding semantic partition tree) of a HTML document. Formally, we have the following definition.

Definition 1 (Types) *Given a DOM tree: (i) Let* $t_1, t_2 \ldots, t_k$ *be the sequence of HTML tags, with their attribute values, on the path from the root of the DOM tree to a leaf node, then* $t_1 \cdot t_2 \ldots t_k$ *is a primitive type; (ii) If* $T_1, \ldots, T_k$ *are types, then* $seq(T_1 \ldots T_k)$ *is a compound type.*

Intuitively, a primitive type encodes the presentation style (including location and visual cues such as font type and size) of a piece of text that corresponds to a leaf node in a DOM tree. For example, in the DOM tree of Figure 3 (which corresponds to the HTML document shown in Figure 1, all the leaf nodes corresponding to the main taxonomic items, “NEWS”, “OPINION”, “FEATURES”, ..., etc., have the same primitive type, $tr \cdot td \cdot table \cdot tr \cdot td \cdot img$. Let us denote this type as T_1. Further observe that all the subtaxonomic items under each main taxonomic item, such as “International”, “National”, ..., etc., under the “NEWS” item, have the same primitive type, $tr \cdot td \cdot table \cdot tr \cdot td \cdot a \cdot font_0$.[4] Let us denote it using T_2.

A compound type essentially summarizes the structural recurrence information at a subtree rooted at an internal node. Note that in Figure 3 the subtree

[4] The $font$ tags with different subscripts in Figure 3 (*e.g.*, $font_0$) denote $font$ tags with different attributes such as type and size.

rooted at the *table* node (shown in circle) groups together several main taxonomic items each of which is followed by a number of subtaxonomic items, *i.e.*, the entire taxonomy is clustered under this *single* DOM tree. This property of spatial locality combined with consistency in presentation style reveals evident structural recurrence information about semantically related items. If we cast all the primitive types of the leaf nodes in the subtree rooted at *table*, then we obtain the following string: $T_1T_2T_2 \ldots T_1T_2T_2 \ldots$. In this string the *sequential pattern*, $T_1T_2^*$ (here $*$ denotes Kleene closure), exactly captures the structural recurrence information of each semantically related item (*i.e.*, a main taxonomic item followed by a number of subtaxonomic items). In our type settings, the sequential pattern $T_1T_2^*$ is *generalized* as the compound type $seq(T_1T_2)$, which is assigned to the type of the *table* node (shown in circle) in Figure 3.

Therefore, as illustrated by the example above, the idea underlying structural analysis is to discover sequential patterns on the type sequence of nodes in a DOM tree. Given any two types as defined above, their equivalence is defined straightforwardly: two types are equivalent if and only if they are syntactically the same. Our structural analysis algorithm is built on the notion of *maximal repeating substrings*, which is defined below.

Definition 2 (Maximal Repeating Substrings) *Given a string S and a support threshold value θ, a substring α that repeats k times in S is a maximal repeating substring if and only if: (i) $k \geq 2$ and $|\alpha| \times k \geq \theta \cdot |S|$ ($|S|$ denotes the length of S); and (ii) $|\alpha| \times k$ is the maximum among all substrings that satisfy condition (i); and (iii) k is the maximum among all substrings that satisfy conditions (i) and (ii).*

Essentially the above definition says that a maximal repeating substring should be, first of all, a repeating substring that covers a majority of elements.[5] In addition, its coverage should be maximized and its length minimized (under the prerequisite that its coverage be maximized). In the sequel, we will use $MaximalRepeatingSubstring(S)$ to represent any algorithm that returns a maximal repeating substring of the input string S if there is one. Otherwise, we assume that it returns the empty string ε.

Since semantically related items exhibit spatial locality, structural analysis can be performed recursively bottom-up starting from the leaf nodes of the DOM tree of a HTML document. Specifically, to transform the DOM tree of a HTML document into a semantic partition tree, we simply invoke the top-level algorithm $PartitionTree$ on the root of the given DOM tree. This algorithm first traverses the DOM tree top-down and then restructures it bottom-up.

[5] The support threshold value is used to filter out "noise" during structural analysis. Since our algorithm traverses a DOM tree bottom-up and multiple semantically related items can be dispersed in several subtrees, the "true" sequential pattern may not be discovered until our algorithm is invoked on an internal node closer to the root. Normally the support threshold value is set to 30% - 50% in our system.

```
Algorithm PartitionTree($n$)
input
    $n$ : a node in a DOM tree
begin
1.  if $n$ is a leaf node then
2.      $n.type$ = the sequence of HTML tags from the root to $n$
3.  else if $n$ has only one child node $c$ then
4.      PartitionTree($c$)
5.      Replace $n$ with $c$ and remove $n$ from the DOM tree.
6.  else
7.      for each child node $x$ of $n$ do PartitionTree($x$) endfor
8.      Analyze($n$)
9.  endif
end
```

We use the notation *n.type* to refer the type attribute of a node n. In the algorithm *PartitionTree*, Line 2 assigns primitive types to all leaf nodes. Internal nodes with only one child are handled in Lines 4–5. In such a case, the type of this only child node is computed and then simply propagated up the tree. However, for an internal node with multiple children, we first invoke *PartitionTree* on all of its children to collect their type information (Line 6). Then the algorithm *Analyze* is invoked upon this node to perform a pattern discovery on its children nodes (Line 7). (Here *Analyze* only does structural analysis. In Section 2.2 we will show how to extend it to do semantic analysis.)

```
Algorithm Analyze($n$)
input
    $n$ : an internal node in a DOM tree
begin
    $S$ = the sequence of all the child nodes of $n$
    for each node $c$ in $S$ do
        if $c.flatten = true$ then
            Replace $c$ with the sequence of all the child nodes of $c$.
        endif
    endfor
    $\tau = \varepsilon$
    do
        Collapse adjacent nodes in $S$ which share the same type.
        $\alpha$ = MaximalRepeatingSubstring(TypeStr($S$))
        if $\alpha \neq \varepsilon$ then $\tau = \alpha$ endif
        if $|\alpha| > 1$ then
            for each substring $\rho$ in $S$ such that TypeStr($\rho$) = $\alpha$ do
                Replace $\rho$ with NewNode($\rho$,$seq(\alpha)$).
            endfor
        endif
    while $|\alpha| > 1$
    if $\tau = \varepsilon$ then
        $n.flatten = true$
    else
        Partition $S$ into $\beta_0\gamma\beta_1 \ldots \gamma\beta_m$, where TypeStr($\gamma$) = $\tau$.
        for each $\gamma\beta_i$ do Replace $\gamma\beta_i$ with NewNode($\gamma\beta_i$, NewType($\tau$)). endfor
        $n.type$ = NewType($\tau$)
    endif
    Make the nodes in $S$ the new children of $n$.
end
```

The algorithm *Analyze* takes an internal node, n, as input. Its main function is to discover structurally similar items among all the children of n and restructure the subtree rooted at n accordingly. Because our algorithm climbs up a DOM tree from leaf nodes to the root, structural similarity may not be discovered until it reaches a node close enough to the root. Therefore, we associate a boolean attribute, $flatten$, with each node to signal whether a structural similarity pattern has been discovered at this node. The value of this attribute is initialized to $false$ for each node. However, if a pattern (or type) is not found at a node, then its $flatten$ attribute is set to $true$ (Line 19).

In Lines 1–6, all the child nodes of n are collected into a sequence, which will be partitioned into semantically related items later if they share structural similarity. But if we encounter a node, c, whose $flatten$ attribute has the value $true$ (which means a pattern is not found at this node), then we move all the child nodes of c into this sequence for further processing. Note that when the algorithm *Analyze* is invoked on a node, all of its descendant nodes are already typed. Intuitively, since the type of a node summarizes the structure of the subtree rooted at that node, analysis of the sequence of sibling types is essential for structural similarity pattern discovery, which is done in two stages by our algorithm.

In the first stage, consecutive nodes having equivalent types are collapsed into a single node (Line 9). The intuition behind this is that they all relate to the same item. Next, in Line 10, an attempt is made to find a maximal repeating substring of the string corresponding to the type sequence of S (returned by $TypeStr(S)$). If such a substring does not exist (hence no structural similarity), then the loop in Lines 8–17 is exited and the $flatten$ attribute of the current node is set to $true$ (Line 19). However, if a maximal repeating substring, α, is found and α contains at least two elements ($|\alpha| > 1$), then the sequence of consecutive nodes whose type sequence matches α is merged into a new node created by the procedure $NewNode$ (Lines 12–16). The first argument of $NewNode$ contains the sequence of nodes to be merged while the second argument indicates the type of this new node. The above collapsing-discovering-merging process is repeated until it cannot be performed any more.

In the second stage (Lines 21–23), the last pattern discovered during the first stage is used to partition the remaining sequence of nodes further. This is a simple heuristic that we apply to handle variations in document structures (*e.g.*, missing data items). Note that if τ contains only one type, then $NewType(\tau)$ returns τ directly; otherwise, it returns the compound type $seq(\tau)$.

Now we illustrate the working steps of the algorithm *Analyze* using an example. For simplicity, we will just show how it manipulates a sequence of types and omit other details. Suppose the type sequence of S is $T_1T_2T_3T_2T_3T_4T_1T_2T_3T_5$ immediately before the algorithm executes the loop starting at Line 8. T_2T_3 is a maximal repeating substring. Let us use a new type T_6 to denote $seq(T_2T_3)$. Then after the first iteration of the loop, the type sequence becomes $T_1T_6T_6T_4T_1T_6T_5$. The first two occurrences of T_6 can be collapsed into one, resulting in $T_1T_6T_4T_1T_6T_5$, in which T_1T_6 is a maximal repeating substring. Again, we use a new type T_7 to represent $seq(T_1T6)$. So after the second iteration the type sequence becomes $T_7T_4T_7T_5$ and the loop terminates. It is not hard to see that the first T_7 and the following T_4 will be put into one partition and the rest into another partition. T_7 is the type assigned to the current node.

2.2 Semantic Analysis

There are two main problems with structural analysis. Firstly, it may not, in general, *always* yield correct partitions corresponding to concept instances, especially when there are structural dissimilarities among them. In particular, the analysis based on maximal repeating substrings (immediately after Line 17 in the algorithm *Analyze*) does not guarantee complete partitions. For example, observe that in Figure 1 the third major headline news item starting with "Bush Backs ..." does not have any pointer to related news while all the others do. Invoking algorithm *Analyze* on the *td* node in Figure 3 (shown in circle which contains all major headline news items) renders a sequence S as follows: $\gamma \cdot T_4 \cdot \gamma \cdot T_4 \cdot \gamma \cdot \gamma \cdot T_4$, where $\gamma = seq(T_1 T_2 T_3)$; T_1, T_2, T_3, T_4 correspond to news title, source, text summary, and pointers to related news, respectively. (The correct partitions corresponding to the four major headline news items should be $P_1 = \gamma \cdot T_4$, $P_2 = \gamma \cdot T_4$, $P_3 = \gamma$, $P_4 = \gamma \cdot T_4$, such that $S = P_1 \cdot P_2 \cdot P_3 \cdot P_4$.)

The second problem with structural analysis is concerned with assigning semantic labels to partitions. Usually the labels of (small) partitions deep in a partition tree are provided by Web site designers in the document itself (*e.g.*, "NATIONAL," "INTERNATIONAL" ..., etc. appearing in the third column in Figure 1). When such a label is present in a document, it is usually the first text item in the partition. Applying this observation as a heuristic, we can extract labels for many partitions. On the other hand, we also have to deal with labeling concept instances using labels not present in the document (*e.g.*, "Major Headline News" in Figure 2). Such situations occur when smaller partitions are aggregated into bigger partitions (*e.g.*, "News Taxonomy" in Figure 2).

To address the problems above, we incorporate semantic analysis techniques into algorithm *Analyze* introduced in Section 2.1. Specifically, Lines 18–24 of algorithm *Analyze*, which serve as a rather straightforward heuristic to finally separate concept instances, will now be replaced with more general techniques for semantic analysis. Due to space limitation, here we will only outline the ideas.

Lexical Association. Lexical association is a light-weight linguistic processing technique for identifying small segments of related text. It semantically relates two *consecutive* pieces of free text by examining whether they share common words (after filtering out "stop" words such as "the" and "of") either directly or via synonym relationship. This process can be implemented using WordNet [1], a widely popular digital lexical reference database.

For instance, recall the example at the beginning of this section, where pure structural analysis produces the sequence $S = \gamma \cdot T_4 \cdot \gamma \cdot T_4 \cdot \gamma \cdot \gamma \cdot T_4$. At this point there is no justification to sort out the correct concept instances, $P_1 = \gamma \cdot T_4$, $P_2 = \gamma \cdot T_4$, $P_3 = \gamma$, $P_4 = \gamma \cdot T_4$, corresponding to the four major headline news items (in the second column of Figure 1) such that $S = P_1 \cdot P_2 \cdot P_3 \cdot P_4$. However, observe that in partition P_4, the texts associated with γ and the following T_4 share the common word "SARS". Therefore, by lexical association γ and T_4 can be semantically related and hence merged into one partition with high confidence.

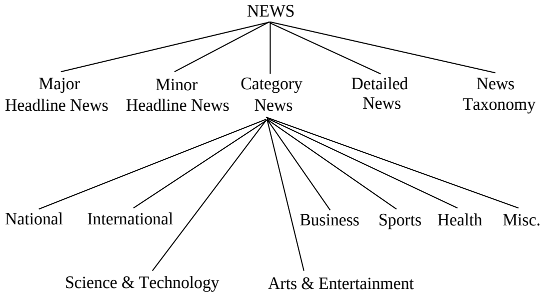


Fig. 4. An Ontology for the News Domain

Concept Association. Concept association maps a partition to a semantic concept that succinctly summarizes the meaning of its content. The concept becomes the label of the partition. To make concept association we leverage domain knowledge encoded in an *ontology.* Informally an ontology describes concepts and their relationships, their features and attributes in a domain of interest. For instance, an ontology for the news domain is shown in Figure 4 and Table 1. Note that all the concepts and their subsumption hierarchy is depicted in Figure 4.

Recall the observation that the labels for (small) partitions may be present in the document itself. Moreover, they are usually the first text item of a partition. This observation leads to the following heuristic for discovering such labels. Suppose a type sequence $T\alpha$ corresponds to a partition. If T consists of short text (one to two words) and matches one of the keywords of a concept in the ontology, then it is extracted as the label for the corresponding partition (*e.g.*, "NATIONAL", "INTERNATIONAL" ..., etc. in the third column of Figure 1).

In cases where labels are not present in the document for the partitions, rules encoded in the ontology are used to classify the content of a partition. For instance, Table 1 shows the concept association rules for the news ontology. To determine if a partition can be classified as a major headline news item the ontology uses a rule which is a function of the key features associated with it, namely, (hyperlink) title, keywords for recognizing news sources (such as AP, Reuters, etc.), and features associated with news summaries such as constraints on the text length.[6]

Propagating Lexical and Concept Associations. In principle, the lightweight semantic analysis techniques introduced above are "incomplete" and hence not all concept instances can be identified. However, recall the key observation that semantically related items exhibit both consistency in presentation style and spatial locality. We can exploit this observation to *propagate* lexical as well as concept associations to other structurally similar items.

[6] More sophisticated classifiers such as Naive Bayes can also be used to make concept associations.

Ontology Concept	Mapping Function
Major Headline News	Rule: Function of Title, Source, Content
Minor Headline News	Rule: Function of Title, Content
National	Keyword: National, U.S.
International	Keyword: International, World
Science & Technology	Keyword: Science, Technology
Arts & Entertainment	Keyword: Arts, Movies, Music, Entertainment, Books, Travel
Business	Keyword: Business, Finance
Sports	Keyword: Sports, Baseball, Basketball
Health	Keyword: Health, Fitness
Detailed News	Rule: Function of Title, Content
News Taxonomy	Keyword: NEWS, OPINION

Table 1. Ontology Concepts and Their Mapping Functions

For instance, recall again the example illustrating lexical association above. In that example we determined via lexical association that γ and T_4 in P_4 are semantically related. Such an association between γ and T_4 can be propagated to other nearby (γ, T_4) pairs to form the partitions P_1 and P_2 (and so the remaining γ becomes P_3).

Structural Types vs. Semantic Types. Like lexical associations concept associations discovered can also be propagated to structurally similar items to assign semantic labels to partitions. In contrast to *structural* types (see Definition 1) that summarize the structural recurrence information about semantically related items, semantic labels can be viewed as *semantic* types that directly capture the semantics of partitions. Because semantic types factor out structural disparities, weaving structural and semantic types together enables our sequential pattern analysis process to uncover higher level concepts in a complex ontology.

Now each node in a DOM tree is associated with two types: a structural type and a semantic type. Once a semantic type (and the semantic label) is assigned to the root node of a partition, it will replace its structural type (if it exists) in the structural analysis process. However, its structural type is still retained in order to propagate concept associations. This will enable structural and semantic analysis to work in tandem. Specifically, immediately prior to invoking structural analysis on the root of a DOM subtree, the semantic type of a child node is propagated to all its siblings having the same structural type.

3 Experimental Results

We have implemented a prototype system and used it to create semantic partition trees from 50 sample HTML documents collected from 8 different news portals. The ontology used in our experiment is shown in Figure 4. These 50 sample HTML documents consist of 15 “front” pages (which contain multiple concepts and concept instances such as news taxonomy and various headline news items) and 35 “detailed” news pages (which are usually characterized by a long text description).

Portal	D	Major Headline News			Minor Headline News			Category News			Detailed News			News Taxonomy		
		N	MF	SD	N	MF	SD	N	MF	SD	N	MF	SD	N	MF	SD
New York Times	15	4	47	5.24	2	28	0.47	19	97	37.32	-	-	-	6	79	27.69
	16	-	-	-	1	16	2.26	-	-	-	1	26	14.88	-	-	-
CNN	14	1	23	5.00	-	-	-	12	13	29.55	-	-	-	1	23	8.04
	13	-	-	-	-	-	-	2	9	5.40	1	32	12.91	1	23	7.87
Yahoo News	18	-	-	-	-	-	-	11	49	49.66	-	-	-	1	24	9.68
	21	-	-	-	3	16	7.05	-	-	-	1	62	22.94	1	30	3.37
Google News	13	26	30	64.16	1	30	4.14	-	-	-	-	-	-	1	15	4.97
	16	-	-	-	2	25	56.30	-	-	-	1	25	20.70	1	34	0.90
ZdNet	13	15	35	12.47	3	6	4.76	-	-	-	-	-	-	-	-	-
	13	-	-	-	4	25	14.39	-	-	-	1	41	18.30	-	-	-
CNet	12	7	79	47.50	3	17	10.29	2	27	6.61	-	-	-	1	7	0
	11	-	-	-	1	22	10.80	-	-	-	1	43	27.26	1	7	0
Bloomberg News	21	7	31	31.10	-	-	-	-	-	-	-	-	-	1	37	5.31
	2	-	-	-	-	-	-	-	-	-	1	30	3.33	1	40	9.80
Recorder News	14	10	25	9.56	-	-	-	-	-	-	-	-	-	1	32	4.06
	5	-	-	-	-	-	-	-	-	-	1	46	1.12	-	-	-

Table 2. Complexity Measures of Sample HTML Documents

To reflect the complexity of a HTML document in terms of its layout structure and content richness, we have collected statistics about all the 50 sample documents (see Table 2). Technically the complexity of a HTML document can be measured along four different dimensions: (1) Mean depth of the DOM tree (denoted by D). A large mean depth value usually implies complicated presentation style; (2) Number of concepts w.r.t. the ontology and their instances (denoted by N). Usually large values are observed on front pages reflecting their rich content; (3) Mean of the maximum fanout of subtrees in the DOM tree that are mapped to ontology concepts (denoted by MF). The maximum fanout of a tree is the max of all the fanout values of descendant nodes. A large value usually implies the presence of multiple concept instances that are clustered together in close vicinity; (4) Standard deviation of structural types of leaf nodes contained in the subtrees that are mapped to ontology concepts (denoted by SD). A larger values usually indicates a higher degree of structural variation among the concept instances.

We compute all the four metrics as defined above for each ontology concept in every HTML document in our sample data set. When each metric is finally tallied, we group together all documents collected from the same news portal and take the mean value of each group. Furthermore, each group is divided into front and detailed news pages and their corresponding mean values of each metric are shown in the top and bottoms rows, respectively, in Table 2 for every news portal. For instance, with respect to the documents from New York Times, the average mean depth of front (detailed) news pages is 15 (16); the average number of “Major Headline News” instances is 4 for front news pages. Note that in Table 2 the symbol “-” indicates that a metric is not applicable.

Portal	Major Headline News		Minor Headline News		Category News		Detailed News		News Taxonomy	
	Rec.%	Prec.%	Rec.%	Prec.%	Rec.%	Prec.%	Rec.%	Prec.%	Rec.%	Prec.%
New York Times	100 -	100 -	50 25	100 100	100 -	100 -	- 100	- 100	100 -	100 -
CNN	100 -	100 -	- -	- -	100 0	100 0	- 100	- 100	100 100	100 100
Yahoo News	- -	- -	- 66.7	- 100	81.8 -	100 -	- 100	- 100	100 100	50 100
Google News	76.9 -	100 -	100 -	100 -	- -	- -	- 100	- 100	100 100	100 100
ZdNet	93.3 -	100 -	66.7 78.9	100 93.3	- -	- -	- 100	- 100	- -	- -
CNet	100 -	100 -	87.5 0	100 0	100 -	100 -	- 100	- 100	100 100	100 100
Bloomberg News	100 -	100 -	- -	- -	- -	- -	- 100	- 100	100 100	100 100
Recorder News	90 -	90 -	- -	- -	- -	- -	- 100	- 100	100 -	100 -

Table 3. Recall and Precision Measures of Semantic Partitions

We measure two performance metrics: *recall* and *precision*. Recall value for a concept in a document is the fraction of the number of partitions correctly labeled as instances of the concept over the actual number of concept instances present in the document. Precision value for a concept is the fraction of the number of partitions correctly labeled as instances of the concept over the total number of partitions (correctly and erroneously) labeled as instances of the concept. To collect recall and precision measures, first we manually identify all the ontology concepts and their correct instances for each document and thus obtain our reference model. When the semantic partition trees are rendered by our algorithm, we manually tally the number of partitions that are correct and erroneous w.r.t. the reference model and thereby compute the recall and precision values. As before, these recall and precision numbers are presented in Table 3 for every news portal and every concept w.r.t. the front and detailed news pages, respectively.

The recall and precision numbers presented in Table 3 show that our algorithm for automatically partitioning HTML documents works well in practice. On detailed news pages which typically contain few concepts and concept instances our algorithm shows high recall and precision. Even for front pages which normally include multiple concepts and instances our algorithm shows little degradation.

It is noteworthy pointing out that our algorithm almost always achieves 100% recall and precision on taxonomic news items, which typically show high structural homogeneity (see the low SD values in Table 2). For some complex pages (such as New York Times and Yahoo front page) the recall and precision for category news is high in spite of their complexity (see SD and MF values of these pages in Table 2 for category news). On the other hand the recall for major headline news in Google's front page is considerably lower than that of CNet's front page because of many more concept instances in Google. Also observe that in ZdNet's front page, in spite of the low complexity of minor headline news (see the low N, MF, and SD values) the recall value is low. This is due to the the "incompleteness" of our semantic analysis techniques.

The running time of our algorithm consists of two components: one is due to structural analysis while the other due to semantic analysis. The complexity of the structural analysis is dominated by $MaximalRepeatingSubstring$ whose worst case time complexity is quadratic in the length of the input sequence. During the bottom-up traversal of the DOM tree this procedure can be invoked at most n times where n is linear in the number of nodes in the DOM tree. The complexity of semantic analysis is dependent on the implementation technology used for doing lexical and concept associations. Currently in our prototype implementation the light-weight concept association rules have worst case time complexity linear in the number of concepts in the ontology.

Finally, we also measure the total running time that our algorithm takes for processing a document. On a Pentium III machine with a 800MHz processor and 256MB memory, the running times range from 60 milliseconds to 6289 milliseconds for documents with 96 nodes to 1709 nodes in their DOM trees.

4 Related Work

The problem studied in this paper is primarily concerned with automatically annotating HTML documents. The essence of this problem boils down to organizing the concepts and concept instances in a HTML document into a (labeled) semantic partition tree. There are a number of areas related to this problem, namely, XML schema discovery [15, 26, 14, 27], schema inference from HTML documents [8, 2], wrapper construction [17, 7, 25], record boundary detection in HTML documents [12, 11, 10, 4], and semantic annotation of HTML documents [18, 19, 9]

However, our approach departs from all the related works above in several respects. Firstly, our main focus is on template-based content-rich HTML documents. Unlike the proposed approaches for XML schema discovery and schema inference from HTML documents, which require a collection of documents as input, we work on *individual* documents. Secondly, our techniques exploit the key observation that semantically related items exhibit consistency in presentation style as well as spatial locality.

Heuristics have been proposed to partition HTML documents into tree-like structures so as to facilitate Web browsing on small-screen devices [5], efficient Web search [33], and converting HTML documents into XML data [6, 32]. However, the partitioning algorithms in these works do not perform complex sequential pattern analysis as proposed by us. The partitioning algorithm of [33] depends on *ad hoc* interpretation of HTML markups and hence does not scale well to arbitray content-rich domains. In [5], a combination of keyword extraction and text summarization is used for semantic partitioning; but the implicit patterns in template-based Web documents are not exploited. The problem of assigning semantic labels to partitions was not addressed in [32]. Unlike our work, [6] did not propose general techniques for discovering semantic labels that are present in a document. Moreover, our work further complements these above works by introducing a novel semantic analysis framework based on discovering and propagating concept associations.

Relevant to our work is the problem of record boundary detection in HTML documents pioneered in [12]. Several heuristics were introduced in [12] for this problem and were later extended in [4]. More recently, a sophisticated framework for developing application ontologies and using them for delineating record boundaries was proposed in [10]. However, these techniques are not directly applicable to our problem due to the presence of rich content comprising multiple concepts and concept instances in one document. Furthermore, like most proposed wrapper construction techniques, [10] requires supervised training whereas our framework does not.

Recently, interesting works were reported in [20, 18, 19, 9], which seek to enable the Semantic Web by enriching Web documents with semantic labels. In [20, 18, 19] powerful ontology management systems form the backbone of systems that support *interactive* annotation of HTML documents. This is in contrast to our approach where annotation is *automatic* given a domain ontology. Even though annotation in [9] is automatic, it does not exploit the key observation that semantically related items exhibit spatial locality in DOM trees of HTML documents. Consequently, their partitioning algorithm may fail to properly identify concept instances in template-generated HTML documents containing multiple concept instances.

Finally, the fundamental difference between our approach and other works is that our framework tightly couples structural and semantic analysis techniques and propagates semantic associations discovered in order to identify concept instances.

5 Conclusion

There is scope for improving both structural and semantic analysis. In structural analysis we can incorporate more robust pattern discovery techniques based on minimum string edit distances. This can easily accommodate structural differences without requiring semantic analysis. We can enhance semantic analysis by incorporating more powerful classification techniques.

The idea of semantic partitioning has important implication to other data management problems. First, it eases the task of formulating (XPath and XQuery) queries to retrieve data from HTML documents. Semantic partitioning can be used to detect semantic changes in document content, an idea recently explored in [24]. Another application is audio-browsable Web content. By putting a dialog interface to the content of a HTML document which is reorganized based on the knowledge of its schema, a user can easily browse its content using voice commands. Audio browsable Web content can greatly expand the reach of the Web to visually challenged individuals. Finally, the idea of semantic partitioning can enable the creation of self-repairing wrappers.

Acknowledgments

This work was supported in part by the NSF grants CCR-0205376, CCR-0311512, and IIS-0072927. The authors would like to thank Wenfang Tan for implementing the algorithms and collecting experimental results.

References

1. WordNet. http://www.cogsci.princeton.edu/˜wn/.
2. A. Arasu and H. Garcia-Molina. Extracting structured data from web pages. In *ACM SIGMOD*, 2003.
3. T. Berners-Lee and M. Fischetti. *Weaving the Web.* Harper San Francisco, 1999.
4. D. Buttler, L. Liu, and C. Pu. A fully automated object extraction system for the world wide web. In *International Confernece on Distributed Computing Systems (ICDCS)*, 2001.
5. O. Buyukkoten, H. Garcia-Molina, and A. Paepcke. Focussed web searching with PDAs. In *International World Wide Web Conference*, 2000.
6. C. Y. Chung, M. Gertz, and N. Sundaresan. Reverse engineering for web data: From visual to semantic structures. In *ICDE*, 2002.
7. W. Cohen, M. Hurst, and L. Jensen. A flexible learning system for wrapping tables and lists in html documents. In *International World Wide Web Conference*, 2002.
8. V. Crescenzi, G. Mecca, and P. Merialdo. Roadrunner: Towards automatic data extraction from large web sites. In *VLDB*, 2001.
9. S. Dill, N. Eiron, D. Gibson, D. Gruhl, R. Guha, A. Jhingran, T. Kanungo, S. Rajagopalan, A. Tomkins, J. Tomlin, and J. Yien. SemTag and Seeker: Bootstrapping the semantic web via automated semantic annotation. In *International World Wide Web Conference*, 2003.
10. D. Embley and L. Xu. Record location and reconfiguration in unstructured multiple-record web documents. In *ACM International Workshop on the Web and Databases (WebDB)*, 2000.
11. D. W. Embley, D. M. Campbell, R. D. Smith, and S. W. Liddle. Ontology-based extraction and structuring of information from data-rich unstructured documents. In *International Conference on Information and Knowledge Management (CIKM)*, 1998.
12. D. W. Embley, Y. Jiang, and Y.-K. Ng. Record-boundary discovery in web documents. In *ACM SIGMOD*, 1999.
13. D. Fensel, S. Decker, M. Erdmann, and R. Studer. Ontobroker: Or how to enable intelligent access to the WWW. In *11th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, Canada, 1998.
14. M. Garofalakis, A. Gionis, R. Rastogi, S. Seshadri, and K. Shim. XTRACT: A system for extracting document type descriptors from xml documents. In *ACM SIGMOD*, 2000.
15. R. Goldman and J. Widom. DataGuides: Enabling query formulation and optimization in semistructured databases. In *VLDB*, 1997.

16. B. N. Grosof, I. Horrocks, R. Volz, and S. Decker. Description logic programms: Combining logic programs with description logic. In *International World Wide Web Conference*, 2003.
17. J. Hammer, H. Garcia-Molina, S. Nestorov, R. Yerneni, M. M. Breunig, and V. Vassalos. Template-based wrappers in the TSIMMIS system. In *ACM SIGMOD*, 1997.
18. S. Handschuh and S. Staab. Authoring and annotation of web pages in CREAM. In *International World Wide Web Conference*, 2002.
19. S. Handschuh, S. Staab, and R. Volz. On deep annotation. In *International World Wide Web Conference*, 2003.
20. J. Heflin, J. A. Hendler, and S. Luke. SHOE: A blueprint for the semantic web. In D. Fensel, J. A. Hendler, H. Lieberman, and W. Wahlster, editors, *Spinning the Semantic Web*, pages 29–63. MIT Press, 2003.
21. I. Horrocks. Benchmark analysis with FaCT. In *TABLEAUX*, 2000.
22. I. Horrocks. DAML+OIL: A description logic for the Semantic Web. *IEEE Bulletin of the Technical Committee on Data Engineering*, 25(1), March 2002.
23. I. Horrocks and S. Tessaris. Querying the semantic web: A formal approach. In *International Semantic Web Conference (ISWC)*, 2002.
24. S.-J. Lim and Y.-K. Ng. A automated change detection algorithm for HTML documents based on semantic hierarchies. In *ICDE*, 2001.
25. L. Liu, C. Pu, and W. Han. XWRAP: An XML-enabled wrapper construction system for web information sources. In *ICDE*, 2000.
26. S. Nestorov, S. Abiteboul, and R. Motwani. Extracting schema from semistructured data. In *ACM SIGMOD*, 1998.
27. Y. Papakonstantinou and V. Vianu. DTD inference for views of xml data. In *ACM PODS*, 2000.
28. M. Sintek and S. Decker. TRIPLE – a query, inference, and transformation language for the semantic web. In *International Semantic Web Conference (ISWC)*, 2002.
29. S. Staab, J. Angele, S. Decker, M. Erdmann, A. Hotho, A. Maedche, H.-P. Schnurr, R. Studerand, and Y. Sure. Semantic community web portals. In *International World Wide Web Conference*, 2000.
30. G. Yang and M. Kifer. $\mathcal{F}$LORA-2: User's Manual. http://flora.sourceforge.net/, June 2002.
31. G. Yang and M. Kifer. Well-founded optimism: Inheritance in frame-based knowledge bases. In *International Conference on Ontologies, Databases, and Applications of Semantics (ODBASE)*, 2002.
32. Y. Yang and H. Zhang. HTML page analysis based on visual cues. In *International Conference on Document Analysis and Recognition (ICDAR)*, 2001.
33. S. Yu, D. Cai, J.-R. Wen, and W.-Y. Ma. Improving pseudo-relevance feedback in web information retrieval using web page segnmentation. In *International World Wide Web Conference*, 2003.