# Automated Metadata and Instance Extraction from News Web Sites

Srinivas Vadrevu, Saravanakumar Nagarajan, Fatih Gelgi, Hasan Davulcu
Department of Computer Science and Engineering,
Arizona State University, Tempe, AZ, 85287, USA
{svadrevu, nrsaravana, hdavulcu, fagelgi}@asu.edu

## Abstract

*In this paper, we present automated techniques for extracting metadata instance information by organizing and mining a set of news Web sites. We develop algorithms that detect and utilize HTML regularities in the Web documents to turn them into hierarchical semantic structures encoded as XML. We present tree-mining algorithms that identify key domain concepts and their taxonomical relationships. We also extract semi-structured concept instances annotated with their labels whenever they are available. We report experimental evaluation for the news domain to demonstrate the efficacy of our algorithms.*

## 1 Introduction

The problem of extracting, managing and organizing the data from unstructured and semi-structured Web pages is an important problem, investigated by several researchers [1, 5, 8]. Critical information such as metadata and attribute labels is usually unlabeled and difficult to locate. It is also presented in various incompatible formats in different Web sources. This data must be digested into an organized into a uniform manner such that it can be used for scalable ad-hoc querying, automatic summarization, integration and mediation over the Web.

There are a plethora of techniques that explore information extraction from semi-structured and unstructured Web sources. For example wrapper induction [10], and semi-automated wrapper learning [2] methods work by learning the path expressions to the data. These approaches require human intervention by either requiring labeled examples or to either manually maintain the wrapper. On the other hand, schema learning [14] and automatic data extraction [5, 1, 11] methods work on structured Web sites to extract the schema and reconstruct the template of the Web pages. These approaches have rigid requirements on the input Web pages that they need to be template-driven and regularly structure their content in an uniform manner. For example, RoadRunner [5] works with a pair of documents from a collection of template generated Web pages to infer a grammar for the collection using union-free regular expressions. Another class of algorithms [16, 3, 6] require that an ontology of concepts, relationships and their value types is provided apriori in order to find matching information.

In order to develop efficient techniques to extract the metadata and instance information from Web pages in an automated manner, it is usually helpful to exploit specific characteristics of the domain of interest. One such domain of interest is that of on-line newspapers and news portals on the Web, which have become one of the most important sources of up-to-date information. There are indeed thousands of sites that provide daily news in a very distinct formats and there is a growing need for tools that will allow individuals to access and keep track of this information in an automatic manner.

In this paper, we present techniques for automatically extracting the metadata and instance information by organizing and mining a set of news Web sites. We extract a common news taxonomy that organizes the important concepts and individual news articles for these concepts with their attribute information.

OntoMiner differs from the earlier information extraction methods in a way that it works in a completely automated manner without any human intervention, it does not require any labeled training examples, and it does not assume anything about the presentation template of the input Web pages. The main contributions of OntoMiner system are threefold, described as following:

- A semantic partitioning algorithm that logically segments the page and groups and organizes the content in an HTML Web page.
- A taxonomy mining algorithm that organizes important concepts in a set of overlapping Web sites.
- An instance mining algorithm that extracts individual instances with their attribute labels from Web pages that belong to the same category.

OntoMiner system is initialized with a collection of news Web sites. It proceeds by detecting and utilizing the HTML

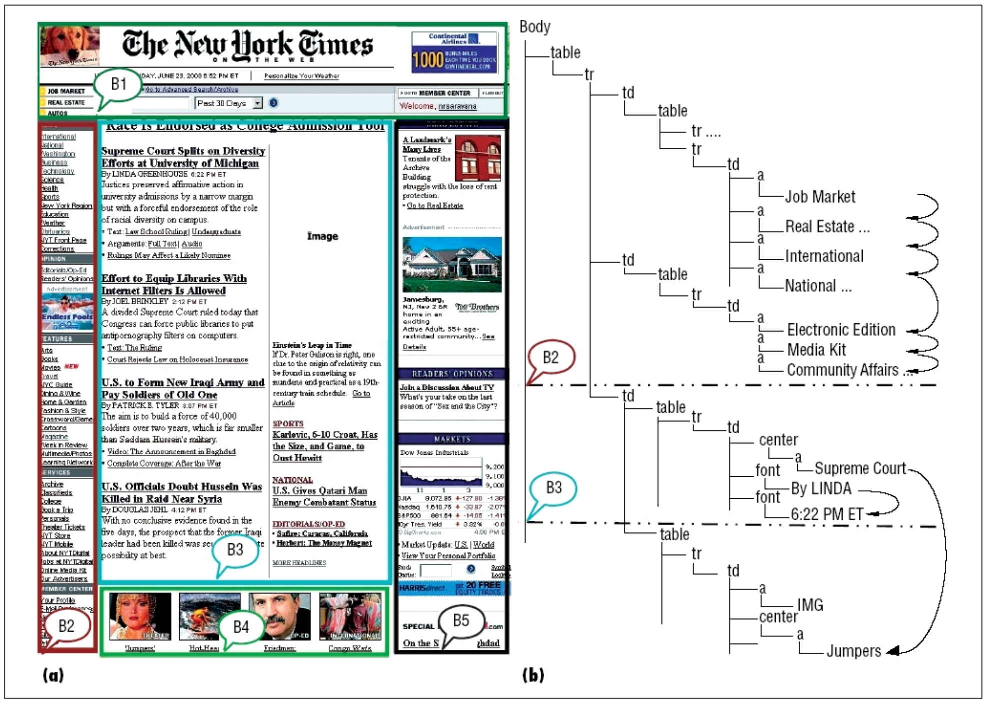

Figure 1. Snapshot of New York Times Home Page

regularities within every Web document and transforming them into hierarchical semantic structures encoded as XML by utilizing a hierarchical partition algorithm. We present tree-mining algorithms that identify most important key domain concepts selected from within the directories of the Home Pages. Next, OntoMiner expands the mined concept taxonomy with sub-concepts by selectively crawling through the links corresponding to key concepts. OntoMiner can accurately separate the "human-oriented decoration" such as navigational panels and advertisement bars from relevant information and utilizes the inferred hierarchical partition corresponding to relevant segments to accurately collect the semi-structured concept instances.

As an example application, a user of the OntoMiner can use the system to rapidly bootstrap an ontology populated with instances and they can tidy-up [7, 15] the bootstrapped ontology to create a rich set of labeled examples that can be utilized by supervised machine learning systems such as the WebKB [4].

Section 2 presents the semantic partitioning algorithm to segment a Web page and organize its content into groups and instances. Sections 3 and 4 describe the taxonomy mining and the news instance extraction algorithms. Section 5 presents the experimental results and Section 6 concludes the paper and describes some of the future directions of this work.

## 2 Semantic Partitioning

OntoMiner employs a two-phase semantic partitioning algorithm made up of flat partitioning, and hierarchical partitioning.

### 2.1 Flat partitioning

Our Flat Partitioner (FP) algorithm detects various logical segments within a Web page. For example, for the homepage of `www.nytimes.com`, we marked the logical segments in boxes B1 through B5 in Figure 1a. The boundaries of segments B2 and B3 correspond to the dotted lines shown in the DOM tree of the Web page in Figure 1b.

FP groups similar contiguous substructures in the Web pages into logical segments by detecting a high concentration of neighboring nodes with similar root-to-leaf tag paths. First, FP initializes the segment boundary to be the first leaf node of the DOM tree. Next, it connects a pair of leaf nodes with a "similarity-link" if they share the same root-to-leaf path and if all other leaf nodes in between have different paths. Then, it calculates the ratio of cardinality of similarity-links crossing the current candidate boundary to that of those within the current segment. If this ratio is less than an experimentally determined threshold $\delta$, set to $0.34$, then FP marks the current node as a segment boundary. Otherwise, it adds the current node to the current segment and considers the next node as a segment boundary.

The process terminates when the algorithm reaches the last leaf node. The tree view in Figure 1b illustrates the FP algorithm.

## 2.2 Hierarchical Partitioning

Hierarchical Partitioning (HP) algorithm infers hierarchical relationships among the leaf nodes of the DOM tree of an HTML page, where all the document content is stored. HP achieves this through a sequence of three operations: binary semantic partitioning, grouping, and promotion.

The binary hierarchical partitioning of a Web page is based on a dynamic programming algorithm that employs the following cost function. It basically creates an hierarchical binary parenthesization of all the leaf nodes yielding a binary partition tree. We recursively define the cost for grouping any two nodes in the DOM tree as follows:

$$Cost(L_i, L_j) = \begin{cases} 0, \quad if\ i = j \\ min_{i \leq k < j}\{Cost(L_i, L_k) + Cost(L_{k+1}, L_j) \\ +Grouping\_Cost(L_{i\ldots k}, L_{k+1\ldots j})\}, \quad if\ i < j \end{cases}$$

where $L_i$, $L_j$ are any two leaf nodes in the DOM tree.

The cost function calculates the measure of dissimilarity between two internal or leaf nodes, that is, a high value of cost indicates that these two nodes' subtrees are highly dissimilar. Thus, the dynamic programming algorithm finds the lowest cost among all possible binary groupings of nodes and parenthesizes them into a binary tree, where similar nodes are grouped together. The cost for grouping two consecutive subtrees is calculated as the sum of five cost factors. Let $A$ and $B$ be the lowest common ancestors (LCA) of nodes $L_i$ to $L_k$ and $L_{k+1}$ to $L_j$, respectively. Then,

Grouping_Cost($L_{i\ldots k}, L_{k+1\ldots j}$) = Grouping_Cost($A$,$B$) = $C_{LCA}(A, B)$ + $C_{PSIM}(A, B)$ + $C_{STSIM}(A, B)$ + $C_{ORD}(A, B)$

The first cost factor $C_{LCA}(A, B)$ calculates how far the two nodes are apart from their LCA. The cost for similarity between paths to the LCA is determined by the second cost factor $C_{PSIM}(A, B)$. The third $C_{STSIM}(A, B)$ and fourth $C_{ORD}(A, B)$ cost factors computes the cost for similarity in the sub trees of the two nodes, former computes the similarity in the paths whereas the later computes the shared ordering of paths in the sub tree. Finally, the fifth cost factor determines how similar are the nodes in the sub trees. OntoMiner calculates the cost factors as shown in Figure 2.

In Figure 3, Column 1 (3a) represents part of the DOM tree of the *New York Times* homepage, and Column 2 (3b) represents the binary hierarchical partition tree. You can see, for example that the algorithm groups nodes 68 through 82 into one partition which has internal binary

1. $C_{LCA}(A, B) = \sqrt{\frac{d_1+d_2}{2*max\_depth}}$

2. $C_{PSIM}(A, B) = 1 - \frac{Similarity\ between\ Paths\ P_1\ and\ P_2}{max(d_1, d_2)}$

3. $C_{STSIM}(A, B) = 1 - max(Separation, Overlap)$,
where $Separation = \frac{|\{S_1 - S_2\} \bigcup \{S_2 - S_1\}|}{|S_1 \bigcup S_2|}$
and $Overlap = \frac{|S_1 \bigcap S_2|}{|S_1 \bigcup S_2|}$

4. $C_{ORD}(A, B) = 1 - Sim(A, B)$,
where Sim(A,B) =

$\frac{Number\ of\ Similar\ Paths\ in\ order\ in\ Sub\ Trees\ of\ A\ and\ B}{Max\ of\ No\ of\ Paths\ in\ Sub\ Trees\ of\ A\ and\ B}$

5. $C_{NSIM}(A, B)$ =0.5 $*$ $depth\_ratio$ + 0.5 $*$ $node\_ratio$,

where $depth\_ratio = \frac{min\_depth}{max\_depth}$,
$min\_depth = min(depth\ of\ A,\ depth\ of\ B)$
$max\_depth = max(depth\ of\ A,\ depth\ of\ B)$

$node\_ratio = \frac{min\_nodes}{max\_nodes}$,
where $min\_nodes = min(|Nodes\ in\ A|,\ |Nodes\ in\ B|)$
$max\_nodes = max(|Nodes\ in\ A|,\ |Nodes\ in\ B|)$

**Figure 2.** Cost Factors for Hierarchical Partitioning

partitions.

**Grouping:** Next, we identify and group a sequence of similar binary partitions under Group nodes. The Group nodes are made up of multiple similar Instance nodes as its children. The grouping algorithm first initializes the type of the leaf nodes in the binary partition tree as "simple". While traversing the tree in post-order, if it finds two "simple" sibling nodes and if the cost for grouping these two nodes is less than a fixed threshold $0.33$ (according to the cost factor evaluation discussed in the previous section), then it marks these nodes as "Instances" and their parent as a "Group" node. Similarly, if it finds two sibling nodes that are marked as "Group" and if the cost for grouping their instances is less than the threshold, then it marks the parent of these sibling nodes as "Group" and merges their "Instances". Alternatively, if one of the sibling nodes is "simple" and the other node is a "Group" node with the above property then the "simple" node is merged with the "Group" node. Figure 3 shows the conversion of a binary partition tree into a Group Tree. The Column 2 and Column 3 represent the binary partition and Group trees respectively.

**Promotion:** The final step in semantic partitioning is *promotion*. The promotion algorithm identifies those leaf nodes that should be promoted above their siblings. Each group structure is labeled with its nearest preceding *emphasized* node and a value instance is labeled with its previous *em-*

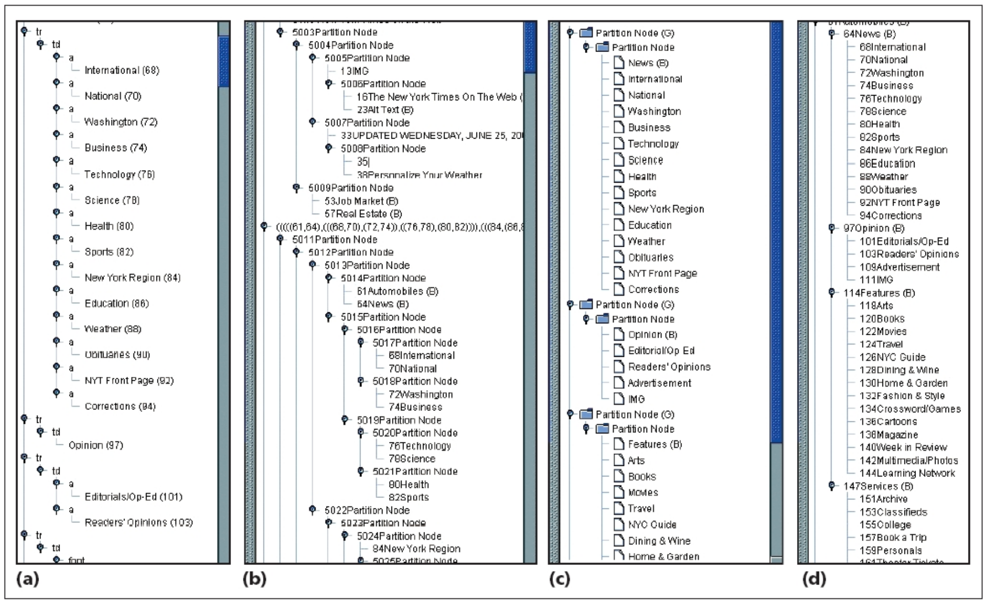

**Figure 3.** Dynamic Programming: (a) Part of the domain object model tree of the *New York Times* home page; (b) its binary partition semantic tree; (c) the converted *Group* tree; (d) the final hierarchical partition tree after promotion.

*phasized* node. A node label is *emphasized* if its labeled text is fully capitalized, or if the html tag of the node has a bold tag.

## 3 Taxonomy Mining

The taxonomy mining involves several tasks, including

- Separating important concepts (the categories that define the context) from instances (the content that are members of each concept), and human-oriented decoration such as navigational panels and advertisement bars, and
- Mining taxonomical relationships among the concepts

Various phases involved in taxonomy mining are explained in the following subsections.

**Mapping Labels to Concepts:** Initially the frequent labels across the input Web sites are obtained and they are preprocessed to eliminate invalid labels (e.g., with no link, and if it points outside the domain, etc.). During this phase, similar concept labels are grouped together based on their lexicographic similarity. The words are stemmed using the Porter's stemming algorithm [17]. Next, Jaccard's similarity coefficient [9], calculated as $\frac{|X \cap Y|}{|X \cup Y|}$, where $X$ and $Y$ are sets of stemmed words from two different labels, is used to group similar labels. We denote each collection of labels to be a *concept* and the labels are recorded using the $Syn$ function. This simple similarity measure groups labels that are lexicographically related (such as "Sport" and "Sports") but not those that are only semantically related (such as "World" and "International"). The issue of identifying and grouping semantically related labels remains to be investigated as future work.

**Mining Taxonomical Relationships:** The concepts obtained from the mapping phase are flat. To organize them into a taxonomy, we need to infer hierarchical relationships among them. Using the algorithm outlined in Algorithm 1, we mine these relationships from the semantically partitioned Web pages.

In this algorithm, $FR$ refers to the bag (a collection of objects whose members need not be distinct) of frequent relationships and $NFR$ refers to the bag of non-frequent relationships. Two concepts $a$ and $b$ in a tree are *i-related* if $a$ is an ancestor of $b$ and $i$ nodes connect them in the tree. We first mine *1-related* pairs, which are direct parent-child relationships and find the frequent relationships. Next, we follow the same procedure for the union of infrequent *1-related* pairs and all *2-related* pairs to find more frequent is-a relationships. We repeat this procedure until we reach the maximum depth of the input trees available for mining. Given the collections of concepts and the hierarchical partition trees corresponding to the relevant home pages, this algorithm produces the concept taxonomy.

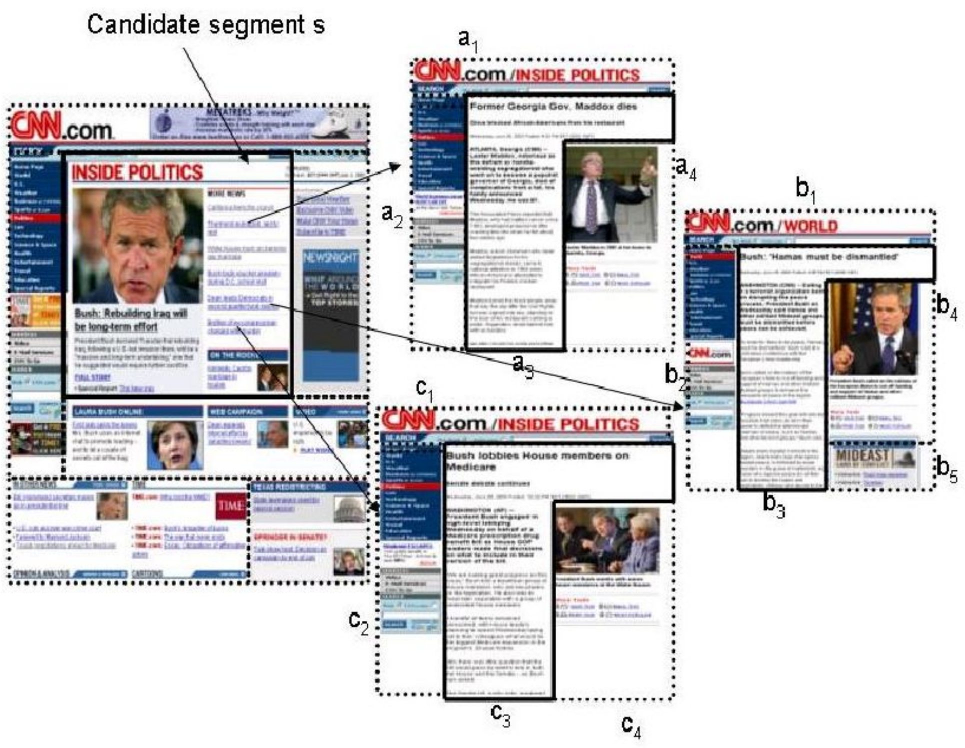

**Figure 4.** Identifying the template of instances from a concept Web page. Solid boxes denote the content of the news articles among instance pages and dashed boxes denote mismatch segments that may correspond to "human-oriented decoration". OntoMiner extracts the individual attributes of the news article from these solid boxes.

**Expanding the Taxonomy beyond Home Pages:** The taxonomy obtained from the previous steps still corresponds to the Home Pages. To expand the domain taxonomy deeper, we follow the links corresponding to every concept label, and expand the taxonomy by repeating the earlier phases, thus identifying sub-concepts. For example, "Sports" is a concept in the taxonomy obtained from the Home Pages. If we follow all the links corresponding to "Sports" concept and repeat the above steps, the sub-taxonomy for the "Sports" concept, that contains sub-concepts such as "Baseball", "Tennis", and "Horse Racing" is obtained. Following the links from all known concepts and mining them yields the final taxonomy for the domain.

**Algorithm 1** Mine Taxonomical Relationships

```
TaxMiner
Input: C: Set of Concepts, S: Set of Semantically Partitioned Web Pages, Sup: Support
Output: Tree representing the hierarchy of concepts

1: $R_0 := \phi$; $NFR_0 := \phi$
2: $d$ := max(depth($s \in S$))
3: for $i$ = 1 to $d$ do
4:     $R_i \leftarrow NFR_{i-1} \bigcup_{bag} i-related\ (S)$
5:     $FR_i$ := frequent($R_i$)
6:     $NFR_i$ := non-frequent($R_i$)
7: end for
8: Return $U_{i=1}^{d}\ FR_i$
```

## 4 News Instance Extraction

A taxonomy schema is made up of a set of hierarchical relationships between concepts. To populate a taxonomy, OntoMiner must identify the concept instances. Instances correspond to members of concepts. In the news domain, the concept instances are the individual news articles and their attributes are title, place, text, place, date, etc.

Instance extraction procedure first identifies the "news article template" from the concept Web page and uses this template to extract the individual news articles by following all the links of that page. The Semantic Partitioning algorithm, of Section 2, is used to arrange the labels in the concept Web page into groups. From each group, the instance extraction procedure fetches any two links and flat partitions them using the algorithm described in Section 2.1 to obtain

| Domain | Number of Pages | **Semantic Partitioner** | | **News Articles Extraction** | |
|---|---|---|---|---|---|
| | | Precision | Recall | Precision | Recall |
| www.abcnews.go.com | 43 | 86% | 89% | 82% | 87% |
| www.cbc.ca | 54 | 89% | 73% | 88% | 74% |
| www.cbsnews.com | 36 | 86% | 89% | 86% | 89% |
| www.cnn.com | 74 | 76% | 84% | 77% | 79% |
| www.foxnews.com | 32 | 93% | 90% | 92% | 93% |
| www.msnbc.com | 48 | 98% | 87% | 91% | 90% |
| www.news.bbc.co.uk | 36 | 83% | 87% | 92% | 89% |
| www.nytimes.com | 64 | 88% | 83% | 97% | 84% |
| www.reuters.com | 37 | 88% | 83% | 86% | 85% |
| www.time.com | 55 | 91% | 90% | 94% | 95% |
| www.timesonline.co.uk | 47 | 90% | 73% | 88% | 71% |
| www.usatoday.com | 29 | 73% | 71% | 75% | 79% |

**Table 1.** Experimental results from a sample 12 Web sites for semantic partitioner and news articles extraction algorithms.

two sequences of logical segments. Next, it aligns these two segment sequences using the Levenshtein distance [12] measure and, Jaccard's content similarity measure is used to identify similar and dissimilar logical segments. The content similarity measure calculates the ratio of the similar words to that of non-similar words.

We argue that the matches in the alignment sequence of the segments of the two news article pages correspond to "human-oriented decoration" such as navigational panels and advertisement bars, where as aligned but dissimilar segments correspond to instance information. The mismatches in the alignment sequence correspond to the actual article content and we extract the individual news article information from these mismatching segments by using the root-to-leaf tag paths of the leaf nodes in them. Such sets of paths are extracted from every group within the concept Web page. The group path set with the most frequently occurring paths among all groups is identified as "news article template". An illustration of extracting the news article template is demonstrated in Figure 4.

Once the segments corresponding to the content instance is identified, instance extraction procedure uses the Semantic Partitioning algorithm to arrange the labels within these segments into a hierarchy. During this phase of the Semantic Partitioning, an additional promotion rule is used that allows the promotion of frequent labels, whenever possible. Once the Semantic Partitions are obtained for the instance segments, the promoted labels are recorded as the attribute labels and their children are recorded as their values.

## 5 Experimental Results

We tested our algorithms on $31$ different news Web sites with a total number of $3216$ individual Web pages. We first present metrics for evaluating the performance of these algorithms and then present the results for the news taxonomy extraction, and news articles extraction algorithms on a sample $12$ news Web sites.

### 5.1 Evaluation

We use the precision and recall metrics for performance evaluations. The precision and recall of semantic partitioning and taxonomy mining algorithms can be calculated by comparing the transitive closure of parent-child relationships inferred by the "algorithmically generated" hierarchies with those implied by the "gold-standard" hierarchies. The "gold-standard" hierarchy for each page and taxonomy was created manually and then the transitive closure of all parent-child relationships were materialized. The precision and recall is calculated by the following

$$Precision = \frac{\{R\}\bigcap\{R'\}}{\{R\}}, \ \ Recall = \frac{\{R\}\bigcap\{R'\}}{\{R'\}}$$

where R and R' are the sets of transitive closure of parent-child relationships implied by the "algorithmically generated" and "gold-standard" hierarchies respectively.

### 5.2 Results

The experimental results for the semantic partitioning and the individual news extraction are as shown in Table 1. The semantic partitioner performs with overall precision of $87\%$ and recall of $83\%$, and the instance extraction algorithm performs with $87\%$ precision and $85\%$ recall respectively. The recall of web pages can be improved by incorporating the Web site specific information so that correct boundaries between segments is detected. Figure 5 shows a fragment of the mined taxonomy for the news domain. The precision and recall values for the generated taxonomy are

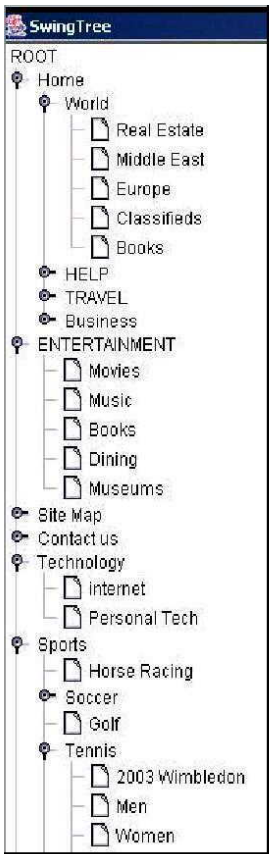

**Figure 5. A fragment of the taxonomy obtained for the News domain**

91% and 79% respectively. The experimental results show that the taxonomy mining algorithm is able to identify the relevant concepts and their labels. The precision and recall for all the algorithms is calculated by manually calculating the "gold-standard" data and comparing to the "algorithmically generated" data.

## 6 Conclusions and Future Work

In this paper, we presented techniques to automatically extract metadata and news instances from news Web sites. The experimental results indicate that our algorithms were able to perform well on various news Web sites. In our future work, we propose to improve the cost factors in order that they can work well for any Web page. We also plan to investigate techniques that combine syntactic as well as semantic regularities [13].

## References

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

[2] N. Ashish and C. Knoblock. Wrapper generation for semi-structured internet sources. *ACM SIGMOD Record*, 26(4), 1997.

[3] F. Ciravegna, S. Chapman, A. Dingli, and Y. Wilks. Learning to harvest information for the semantic web. In *Proceedings of the 1st European Semantic Web Symposium*, Heraklion, Greece, 2004.

[4] M. Craven, D. DiPasquo, D. Freitag, A. K. McCallum, T. M. Mitchell, K. Nigam, and S. Slattery. Learning to extract symbolic knowledge from the World Wide Web. In *Natl. Conf. on Artificial Intelligence*, pages 509–516, Madison, US, 1998. AAAI Press, Menlo Park, US.

[5] V. Crescenzi, G. Mecca, and P. Merialdo. Roadrunner: Towards automatic data extraction from large web sites. In *Intl. Conf. on Very Large Data Bases*, 2001.

[6] S. Dill, J. A. Tomlin, J. Y. Zien, N. Eiron, D. Gibson, D. Gruhl, R. Guha, A. Jhingran, T. Kanungo, S. Rajagopalan, and A. Tomkins. Semtag and seeker: Bootstrapping the semantic web via automated semantic annotation. In *Twelth International Conference on World Wide Web*, pages 178–186, 2003.

[7] A. Doan, P. Domingos, and A. Halevy. Learning to match the schemas of data sources: a multistrategy approach. *Machine Learning*, 2003.

[8] 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.

[9] R. Korfhage. *Information Storage and Retrieval*. John Wiley Computer Publications, New York, 1999.

[10] N. Kushmerick, D. S. Weld, and R. B. Doorenbos. Wrapper induction for information extraction. In *IJCAI*, pages 729–737, 1997.

[11] K. Lerman, L. Getoor, S. Minton, and C. Knoblock. Using the structure of web sites for automatic segmentation of tables. In *ACM SIGMOD*, 2004.

[12] V. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Daklady*, 10:707–710, 1966.

[13] S. Mukherjee, G. Yang, and I. Ramakrishnan. Annotating content-rich web documents: Structural and semantic analysis. In *International Semantic Web Conference*, 2003.

[14] S. Nestorov, S. Abiteboul, and R. Motwani. Extracting schema from semistructured data. In *SIGMOD*, pages 295–306, New York, NY, USA, 1998. ACM Press.

[15] N. Noy and M. Musen. Prompt: Algorithm and tool for automated ontology merging and alignment. In *Proceedings of AAAI-2000, 17th Conference of the American Association for Artificial Intelligence*. AAAI Press, Menlo Park, US.

[16] M. Perkowitz, R. B. Doorenbos, O. Etzioni, and D. S. Weld. Learning to understand information on the internet: An example-based approach. *Journal of Intelligent Information Systems*, 8(2), 1997.

[17] M. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.