

# OntoMiner: Bootstrapping Ontologies From Overlapping Domain Specific Web sites

Hasan Davulcu, Srinivas Vadrevu, Saravanakumar Nagarajan  
Department of Computer Science and Engineering  
Arizona State University  
Tempe, AZ 85287-5406, USA  
{hdavulcu, svadrevu, nrsaravana}@asu.edu

## ABSTRACT

In this paper, we present automated techniques for bootstrapping and populating specialized domain ontologies by organizing and mining a set of relevant overlapping Web sites provided by the user. We develop algorithms that detect and utilize HTML regularities in the Web documents to turn them into hierarchical semantic structures encoded as XML. Next, 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. Experimental evaluation for the News, Hotels, Product and Biology domains indicates that our algorithms can bootstrap and populate domain specific ontologies with high precision and recall.

## Keywords

Web Mining, Semantic Web, Ontology, Data Mining

## 1. INTRODUCTION

In order to enable widespread usability for the Semantic Web there is a need to bootstrap large, rich and up-to-date domain ontologies that organizes most relevant concepts, their relationships and instances. In this paper, we present automated techniques for bootstrapping and populating specialized domain ontologies by organizing and mining a set of relevant overlapping taxonomy-directed Web sites provided by the user. A Web site is said to be "taxonomy-directed" if it contains at least one taxonomy for organizing its key concepts and it presents the instances belonging to each concept in a regular fashion. Notice that, neither the presentation of the taxonomy among different pages, nor the presentation of instances among for different concepts need to be regular for a Web site to be classified as "taxonomy-directed". Almost all scientific, news, financial, travel, shopping and search/community portals that we are aware of are indeed "taxonomy directed".

The user of the OntoMiner system only need to provide the system with the URLs of the Home Pages of 10 to 15 taxonomy-directed overlapping domain specific Web sites that characterizes her domain of interest. A pair of Web sites are said to be *overlapping* if their taxonomies share some concept labels. Next, OntoMiner system detects and utilizes the HTML regularities within every Web document and turns them into hierarchical semantic structures encoded as XML by utilizing a hierarchical partition algorithm. OntoMiner uses tree-mining algorithms to identify most important key domain concepts and relationships among them se-

lected from within the directories of the Home Pages. OntoMiner proceeds with expanding the mined concept taxonomy with sub-concepts by selectively crawling through the links corresponding to key concepts. OntoMiner also has algorithms that can identify the logical regions within Web documents that contains links to instance pages.

A key characteristic of OntoMiner is that, unlike the early pioneering systems described in [1, 2, 3] which attempt to extract data values alone, OntoMiner extracts the labels corresponding to categories and attribute names along with their data values and organizes them into an ontology. We define an *ontology* to be a 7-tuple,  $\mathcal{O} = \langle \mathcal{L}, \mathcal{C}, \mathcal{T}, \mathcal{I}, \mathcal{A}, \text{Syn}, \text{Member} \rangle$ , where

- $\mathcal{L}$  is a set of labels denoting concepts and attributes
- $\mathcal{C}$  is a set of concepts
- $\mathcal{T}$  is a taxonomy relationship representing an is-a hierarchy among concepts,  $\mathcal{T} \subseteq \mathcal{C} \times \mathcal{C}$  is reflexive, asymmetric and transitive
- $\mathcal{I}$  is a set of instances
- $\mathcal{A}$  is a set of attributes, where each attribute is a function that maps instances to labels,  $\mathcal{A} : \mathcal{I} \rightarrow \mathcal{L}$
- $\text{Syn}$  is a function that maps concepts and attributes to their various labels,  $\text{Syn} : \{\mathcal{C} \cup \mathcal{A}\} \rightarrow \mathcal{L}$
- $\text{Member}$  is a function that represents member-of relationship between instances and concepts,  $\text{Member} : \mathcal{I} \rightarrow \mathcal{C}$

## 2. ARCHITECTURE & ALGORITHM

The architecture of OntoMiner is as shown in Figure 1. As illustrated in the figure, OntoMiner analyzes a collection of overlapping domain specific Web sites and generates a taxonomy of important concepts and their associated instances. As the Crawler fetches the Web pages, they are fed to Semantic Partitioner. Semantic Partitioner examines the structure of the HTML Web page and converts them into semantic trees inferring the hierarchy presented in the visual presentation among various labels in the HTML Web page. Taxonomy Miner examines the semantic trees generated by Semantic Partitioner, finds the frequent labels and groups them into concepts  $\mathcal{C}$ . Next, it mines the taxonomy of concepts,  $\mathcal{T}$  using frequent tree mining algorithms. For each concept in the mined taxonomy, Instance Miner identifies its potential instance Web pages and mines instances,  $\mathcal{I}$  consisting of labeled and unlabeled attributes and their values. It extracts the attribute labels  $\mathcal{A}$  and their values by working with two documents, aligning the content in them and extracting the dissimilar content.

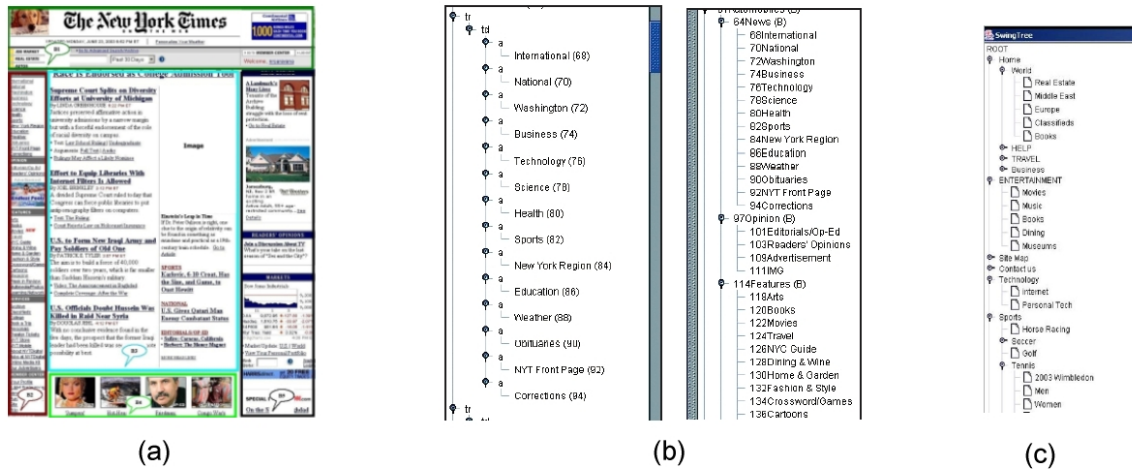


Figure 2: (a) A snapshot of New York Times Front Page with logical segmentation illustrated; (b) Output of the Semantic Partitioner, showing the document object model of the html page and the final hierarchical tree; (c) A fragment of the mined taxonomy of concepts from a collection of News Web sites.

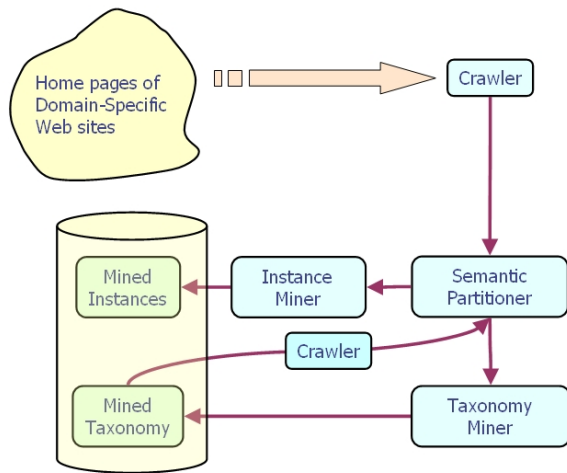


Figure 1: Architecture of OntoMiner

### 3. AN ILLUSTRATIVE EXAMPLE

The example shown in Figure 2 demonstrates the working of the OntoMiner algorithm in several phases. The semantic partitioning algorithm works with single Web page at a time to extract hierarchy among the labels in the Web page. It works by first flat partitioning the Web page into logical segments as shown in Figure 2a. Next, it uses dynamic programming techniques to organize the labels in the Web page into groups that exploit the structural similarities in the Document Object Model (DOM) tree. Finally it uses promotion rules that are based on the presentation and the format of the Web page to promote the emphasized labels on top of certain groups. Figure 2b shows the segment of the DOM tree of the Web page and the final hierarchical tree. OntoMiner employs semantic partitioning algorithm to obtain hierarchical trees on a collection of Web sites and uses taxonomy mining algorithm on these trees to mine the taxonomy of important concepts. The taxonomy mining algorithm uses frequent tree pattern mining techniques to mine is-a

relationships. Figure 2c shows a fragment of the mined taxonomy from a collection of News Web sites.

### 4. CONCLUSIONS AND FUTURE WORK

We present algorithms to logically partition a Web page and infer the hierarchy among the labels in the Web page, mine the taxonomy of important concepts from overlapping domain specific Web sites, extract attributed instances and their values corresponding to each concept in the taxonomy. The novel cost factors used in semantic partitioning algorithm help to infer the logical hierarchical organization of the content within single Web page. For details refer to [4]. OntoMiner only requires a collection of domain specific sites in building the taxonomy of concepts as it uses frequency based techniques to mine the is-a relationships. OntoMiner can find complex semi-structured instances for attributed concepts and can extract labels for attributes whenever they are present. Overall, OntoMiner attains a precision of 90 percent and recall of more than 80 percent for semantic partitioning, taxonomies, attributes and values of their instances. Some of the future directions for our work include investigating techniques that combine syntactic as well as semantic regularities and we will try to iteratively repair hierarchical partitions with bootstrapped ontologies to yield higher precision and recall.

### 5. REFERENCES

- [1] Valter Crescenzi, Giansalvatore Mecca, and Paolo Merialdo. Roadrunner: Towards automatic data extraction from large web sites. In *Proceedings of 27th International Conference on Very Large Data Bases*, pages 109–118, 2001.
- [2] A. Arasu and H. Garcia-Molina. Extracting structured data from web pages. In *ACM SIGMOD*, 2003.
- [3] Christina Yip Chung, Michael Gertz, and Neel Sundaresan. Reverse engineering for web data: From visual to semantic structures. In *Intl. Conf. on Data Engineering*, 2002.
- [4] Hasan Davulcu, Srinivas Vadrevu, and Saravanakumar Nagarajan. Ontominer: Bootstrapping and populating ontologies from domain specific web sites. *IEEE Intelligent Systems*, 18(5), September 2003.