Semantic Partitioning of Web pages

Srinivas Vadrevu, Fatih Gelgi, and Hasan Davulcu

Department of Computer Science and Engineering,
Arizona State University,
Tempe, AZ, 85287, USA
{svadrevu, fagelgi, hdavulcu}@asu.edu

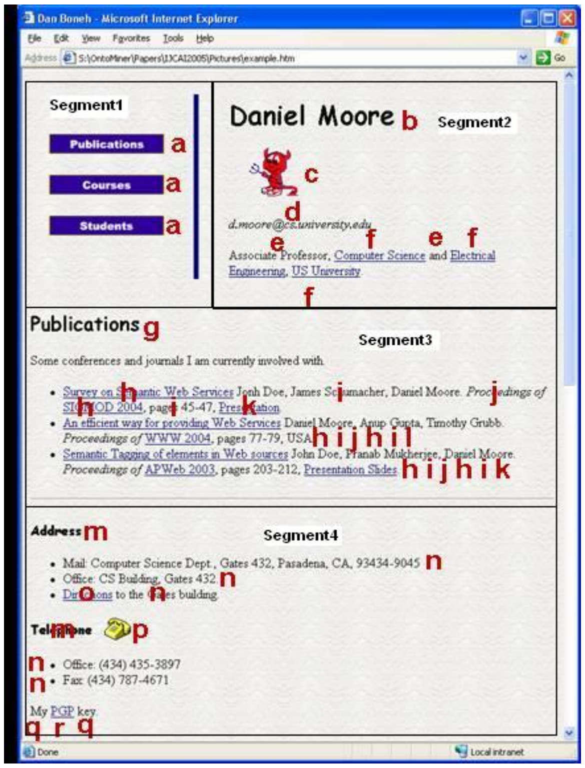
Abstract. In this paper we describe the *semantic partitioner* algorithm, that uses the structural and presentation regularities of the Web pages to automatically transform them into hierarchical content structures. These content structures enable us to automatically annotate labels in the Web pages with their semantic roles, thus yielding meta-data and instance information for the Web pages. Experimental results with the TAP knowledge base and computer science department Web sites, comprising 16, 861 Web pages indicate that our algorithm is able gather meta-data accurately from various types of Web pages. The algorithm is able to achieve this performance without any domain specific engineering requirement.

1 Introduction

Scalable information retrieval [1] based search engine technologies have achieved wide spread adoption and commercial success towards enabling access to the Web. However, since they are based on an unstructured representation of the Web documents their performance in making sense of the available information is also limited.

In this paper we present automated algorithms for gathering meta-data and their instances from collections of domain specific and attribute rich Web pages. Our system works without the requirements that (i) the Web pages need to share a similar presentation template or (ii) that they need to share the same set of meta-data among each other. Hence, we can not readily use previously developed wrapper induction techniques [3, 2] which require that the item pages should be template driven or the ontology driven extraction techniques [5, 6, 4] which require that an ontology of concepts, relationships and their value types is provided apriori in order to find matching information.

Consider the faculty home page in the example shown in Figure 1 that lists the attribute information in a detailed manner. Some of the attributes presented in the Web page are publications, address and telephone. Our system utilizes the presentation regularities within the Web page to annotate the labels in the Web page with their semantic roles, such as Concept (C), Attribute (A) and Value (V). This kind of annotation helps us in separating and gathering meta-data from Web pages, yielding to a powerful tool that can be utilized in various applications.


Dan Boneh - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address S:\OntoMiner\Papers\IJCAI2005\Pictures\example.htm
Go
Segment1
Segment2
Segment3
Segment4
a
a
a
b
c
d
e
f
e
f
f
g
h
h
i
i
j
k
h i j h i l
h i j h i k
m
n
n
o
n
m
p
n
n
q r q
Done
Local intranet

Publications

Courses

Students

Daniel Moore

d.moore@cs.university.edu

Associate Professor, Computer Science and Electrical Engineering, US University.

Publications

Some conferences and journals I am currently involved with.

- Survey on Semantic Web Services Jonh Doe, James Schumacher, Daniel Moore. *Proceedings of* SIGMOD 2004, pages 45-47, Presentation
- An efficient way for providing Web Services Daniel Moore, Anup Gupta, Timothy Grubb. *Proceedings of* WWW 2004, pages 77-79, USA
- Semantic Tagging of elements in Web sources John Doe, Pranab Mukherjee, Daniel Moore. *Proceedings of* APWeb 2003, pages 203-212, Presentation Slides

Address

- Mail: Computer Science Dept., Gates 432, Pasadena, CA, 93434-9045
- Office: CS Building, Gates 432
- Directions to the Gates building

Telephone

- Office: (434) 435-3897
- Fax: (434) 787-4671

My PGP key.

Fig. 1. An example of a Faculty home page. The labels in the page are marked with their corresponding path identifier symbols. The page segments in the Web page are also marked as Segments 1 to 4.

Unlike plain text documents, Web pages organize and present their content using hierarchies of HTML structures. Different logical blocks of content, such as taxonomies, navigation aids, headers and footers as well as various other information blocks such as indexes, are usually presented using different HTML structures. Furthermore, whenever an information block contains a list of items, then these items themselves are presented consecutively and regularly using repeating similar HTML substructures. Our semantic partitioning algorithm [7] can detect such repeating HTML substructures and organize the content into a *hierarchical content structure* consisting of *groups* of *instances*, while skipping HTML blocks that do not constitute a group. Our algorithm is *robust* in a well defined sense; it can accurately identify all groups and their instances even in the presence of certain presentation irregularities. The semantic partition algorithm requires no training and works automatically on each Web page.

The hierarchical organization of the content a Web page is useful in many scenarios. We can use such organization of the content annotate the labels of the page with semantic roles such as Concept (C), Attribute (A) and Value (V) by utilizing the context in which the Web page is obtained. For example, if we can organize the content in a Faculty home page, we would be able identify 'contact information', 'research interests', 'teaching', etc. as attributes of the concept 'Faculty', and we can identify the instances of these attributes as value labels. Such kind of semantic tagging can then be used to empower an information retrieval system that can reason with the context and semantic labels, besides the plain bag of words. We can also use the semantically tagged Web pages to materialize a specialized information retrieval system for specific domains such as sports, restaurants, shopping and educational domains.

The key contributions and innovations of our semantic annotation algorithm can be described as follows:

– It performs extraction without the assumption that the input Web pages are template-driven. For example, we present experimental results with computer science faculty and course home pages which are not template-driven.
– It is domain independent and it does not require a previously engineered domain ontology.
– It effectively extracts and separates meta-data from the instance information in Web pages in a completely automatic fashion.

We currently do not align the extracted meta-data or instance information with any available ontology or knowledge structure. Various existing approaches developed for mapping and merging ontologies [8] can be utilized for this purpose. We also currently do not process plain text inside the Web pages, i.e., any text fragments that contain modal verbs. However, we hypothesize that the meta-data extracted from the attribute rich segments of the Web pages can be used to extract information from text with the help of natural language processing techniques [9].

The rest of the paper is organized as follows. Section 2 gives an overview of the related work. Sections 3, 4, 5, and 6 explain several phases in our semantic partitioning algorithm. Section 7 presents complexity analysis of our algorithms and Section 8 provides experimental results on various data beds. Finally we conclude in Section 9 with some future directions of our work.

2 Related Work

In this section, we present an overview of the related work from several areas and show how our system is different from them.

Template based algorithms: RoadRunner [10] 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. ExAlg [11] is another system that can extract data from template generated Web pages. ExAlg uses equivalence classes (sets of items that occur with the same frequency in every page) to build the template for the pages by recursively constructing the page template starting from the root equivalence class. TAP [12] is a system that extracts RDF triplets from template driven Web sites in order to generate a huge knowledge base that has a Web searchable interface. These algorithms are based on the assumption that the input Web pages are template driven in their presentation and are usually driven by standard meta-data. Our approach differs from all these approaches in that it does not require that the input Web pages are template driven and it can effectively handle noise.

Grammar induction based algorithms: Grammar induction based systems employ a strong bias on the type and expected presentation of items within Web pages to extract instances. Such assumptions like "product descriptions should reside on a single line" [13] and "items may not have missing or repeating attributes" do not apply for most of the Web sites. XTRACT [14] is such a system that can automatically extract Document Type Descriptors (DTDs) from a set of XML documents. It transforms each XML document as a sequence of identifiers and infers a common regular expression that serves as a DTD, using the Minimum Description Length (MDL) principle. Our pattern mining algorithm is different from these approaches and parses the given sequence in a bottom-up fashion and infers the grammar on-the-fly as it goes through the sequence multiple number of times.

Page Segmentation algorithms: VIPS algorithm [15] is a vision-based page segmentation algorithm that is based on HTML based heuristics to partition the page into information blocks. Our page segmentation algorithm is similar to the VIPS algorithm in traversing the DOM (Document Object Model [1]) tree in top-down fashion, but our algorithm uses well-defined information theoretic methods in order to measure the homogeneity of the segment, whereas the VIPS algorithm is based on HTML heuristics.

3 Page Segmentation

A Web page usually contains several pieces of information [15] and it is necessary to partition a Web page into several segments (or information blocks) before organizing the content into hierarchical groups. In this section we describe our page segmentation algorithm that partitions given a Web page into flat segments.

The page segmentation algorithm relies on the DOM tree representation of the Web page and traverses it in a top-down fashion in order to segment the content of the page, which lies at the leaf nodes. We define a segment as a contiguous set of leaf nodes within a Web page. The algorithm aims to find *homogeneous segments*, where the presentation of the content within each segment is uniform. The algorithm employs a split-traverse based approach that treats all uniform set of nodes as homogeneous segments and further splits non-uniform nodes into smaller segments until each segment is homogeneous. The algorithm relies on the concept of *entropy* in order to determine whether a segment is homogeneous.

We define the notion of entropy of a node in a DOM tree in terms of the uncertainty in the root-to-leaf paths under the node. Our algorithm is based on the observation that a well organized or homogeneous system will have low

[1] `http://www.w3.org/DOM/`

entropy. We use this principle while traversing the DOM tree from root to leaf nodes in a breadth-first fashion, and split the nodes until each and every segment in the DOM tree is homogeneous.

Entropy is generally used to measure the uncertainty in the system. Hence if any random variable has low entropy, then there is less uncertainty in predicting the possible values that the random variable can take. In our case, we view each node as a random variable in terms of the root-to-leaf paths P_is under the node. We define a set of nodes in the DOM tree to be homogeneous if the paths in the random variable are uniformly distributed, and it is easy to predict the next path within the set.

Definition 31 *The path entropy $H_P(N)$ of a node N in the DOM tree can be defined as*

$$H_P(N) = -\sum_{i}^{k} p(i)\log p(i),$$

where $p(i)$ is the probability of path P_i appearing under the node N.

We use the concept of path entropy to partition the DOM tree into segments. The algorithm to partition a given Web page into segments is given in Algorithm 1. This algorithm is initialized with a vector of nodes containing just the root node of the DOM tree of the Web page. The $MedianEntropy$ is calculated as the median of the path entropies of all the nodes in the DOM tree. Essentially we assume the nodes whose path entropy is less than the median entropy of all the nodes in the DOM tree to be homogeneous and output it as a pure segment. Otherwise, we traverse the children of the nodes in order to find the homogeneous segments. Our page segmentation algorithm is able to identify four segments in the Faculty home page as shown in Figure 1. Please note that we currently ignore any text fragments in the Web page that contains modal verbs as they add to the noise in identifying the patterns.

Algorithm 1 Page Segmentation Algorithm

```
PageSegmenter
Input: Nodes[], a node in the DOM tree
Output: A set of segments

1: for Each Subset $S$ of $Nodes[]$ do
2:    $H_P(S)$ := Average Path Entropy of all nodes in $S$
3:    if $H_P(S) \leq MedianEntropy$ then
4:       Output all the leaf nodes under $N$ as a new segment $PS$
5:    else
6:       PageSegmenter(Children($S$))
7:    end if
8: end for
```

4 Inferring Group Hierarchy

Even though the page segmentation algorithm organizes the content in a Web page in terms of segments that present the content in a uniform fashion, each segment just contains a flat representation of the labels. In many scenarios, the content is usually represented a hierarchy of labels and we need to infer the hierarchy among the labels in order to properly depict the content structure. We define a *group* to be a contiguous collection of instances that are presented together in a Web page, where an *instance* is a repeating element of a group. Such a group hierarchy is helpful in organizing the content and determining the most general concept in the page and its attributes.

One possible way to achieve such hierarchical organization of the content is to work directly on the DOM tree in a top-down fashion [16]. But such approaches suffer from handling the noise in the leaf nodes and in successfully detecting the boundaries as look-ahead searching is expensive. An efficient alternative approach is to utilize the presentation information (embedded in their root-to-leaf tag path in the DOM tree) of the labels in order to infer a hierarchy among them. We transform each segment into a sequence of path identifiers of root-to-leaf paths of the leaf nodes and infer regular expressions from this sequence in order to extract patterns from them. For example, the path sequence corresponding to the Segment3 as marked in Figure 1 is $ghijhikhijhilhijhik$.

After the segment is transformed into a sequence of path identifiers, we extract patterns from it by inferring a regular expression. Our notion of regular expression contains four special operators: concatenation (.), kleene star (*), optional (?) and union (|). We build the regular expression for a given sequence by incrementally building it using bottom-up parsing. We go through the sequence several times, each time folding it using one of the four operators, until there are no more patterns left. The algorithm for building the regular expression from a sequence is described in Algorithm 2. The *InferRegEx* method is initialized for each segment in the page with its corresponding path sequence. Our algorithm is able to identify the regular expression $g(hijhi(k|l))*$ for the Segment3 in the example shown in Figure 1.

Once the regular expression is inferred from the sequence of path identifiers, every kleene star in the regular expression is treated as a group and its members are treated as instances of the group. For example, in Segment3 of the Web page in Figure 1, the regular expression $(hijhi(k|l))*$ is identified as a group which corresponds to the publications of the corresponding faculty member. The nested kleene star symbols are transformed into nested group hierarchies from the path sequence.

5 Promotion

After grouping, the content of the Web page is organized into hierarchical group structures, but each of these group structures do not have any label. The labels for the groups, which we call as *group headers*, are important as they play a role

Algorithm 2 Inferring Regular Expressions

```
InferRegEx(S)
Input: $S$, a sequence of symbols
Output: $S$, a new regex sequence of symbols
 1: patternFound = true;
 2: repeat
 3:    patternFound = false;
 4:    for len = 1:length($S$)/2 do
 5:       for i = 0:length($S$) do
 6:          currPattern := subseq(j, j+i);
 7:          if ExtendPattern(currPattern, $S$, j+i+1) = true then
 8:             ReplacePattern(currPattern, $S$);
 9:             patternFound = true;
10:          end if
11:       end for
12:    end for
13: until patternFound = true
14: return $S$;
End of InferRegEx
```

```
ExtendPattern(P, S, startIndex)
Input: $P$, current pattern; $S$, a sequence of symbols; startIndex, the start index to look
patterns for
Output: boolean, indicating whether the pattern is extended
 1: for i = startIndex:length($S$) do
 2:    consensusString := IsMatch(currPattern, subseq(i,i+length(currPattern));
 3:    if consensusString $\neq$ null then
 4:       currPattern := consensusString;
 5:       ExtendPattern(currPattern, $S$, i+length(currPattern))
 6:    end if
 7: end for
End of ExtendPattern
```

```
IsMatch($P_1$, $P_2$)
Input: $P_1$, first pattern; $P_2$, second pattern
Output: consensusString obtained by alining $P_1$ and $P_2$ or null
 1: if EditDistance($P_1$, $P_2$) $\leq \frac{MaxLength(P_1,P_2)}{3}$ then
 2:    return Consensus($P_1$, $P_2$)
 3: else
 4:    return null
 5: end if
End of IsMatch
```

```
ReplacePattern(P, S)
Input: $P$, current pattern; $S$, a sequence of symbols
Output: none
 1: replace all occurrences of the pattern $P$ in the sequence $S$ by a new symbol $P'$.
End of ReplacePattern
```

in connecting the repeating set of instances with the corresponding concept or attribute label. For example, in a news Web page such as CNN, a group structure containing all the scientific articles cannot be of much use unless it is labeled as *Science* or *Sci/Tech* for various purposes such as searching.

Bootstrapping the Promotion: In the grouping phase, all the leaf nodes that appear before the group are identified as candidate group headers and the goal of the promotion algorithm is to select the appropriate group header from these candidates for the group. These group headers are bootstrapped by promoting the label as the header whenever there is only one candidate for a particular group.

Frequent Label based Promotion: Whenever similar Web pages from the same domain are available, we identify all the frequent labels in the domain from these similar pages and promote the closest frequent label that is present in the candidate headers of a group as the label for the group.

Naive Bayes based Promotion: When many similar Web pages obtained from similar domains and from the same context are available, the candidate group headers can be used as a training data when deciding a label to promote as a group header for a group. In such scenarios, we use the words in the instances as features to train a Naive Bayes classifier and compute the likelihood of every candidate group header with a set of instances. Later we promote the closest one as the header for the group.

Path Consistent Labeling: Usually similar presentation templates are used to represent similar labels within the same Web page. Hence if one of those similarly presented labels is promoted with the above rules, then we promote all the other labels within the same Web page with the same presentation template on top of the next groups, whenever applicable.

6 Meta-Tagging

After the group hierarchy has been found and the appropriate labels have been promoted on the groups, we have a powerful content structure that organizes the labels in the Web page in a uniform fashion. We utilize this content structure to annotate the labels in the Web page with meta-data tags.

In our annotation framework, we used four different tags: Concept (C), Attribute (A), Value (V), and None (N). A concept label is an abstract or symbolic tag that generally defines the context for a Web page. An attribute label is a property of a concept or the name of the relationship. A value label is an instance of the attribute relationship of a particular concept and a none label is tag that we use to denote a label that does not belong to any of these categories. The Figure 2 shows an example of the assignment of tags in the context of Faculty home pages. Within a certain context concept, we interpret all the group headers as attribute labels and the all the instance values as value labels of the corresponding attribute.

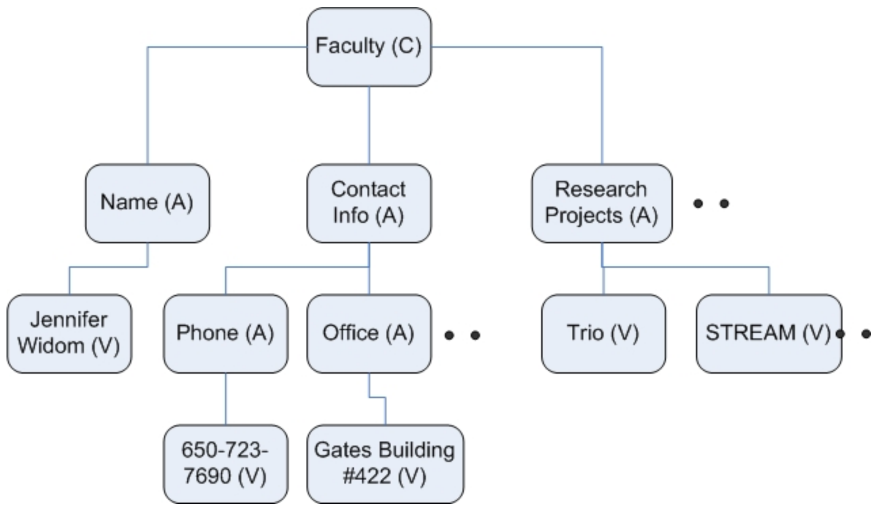


Fig. 2. An example illustrating the meta tagging in a faculty home page.

7 Complexity Analysis

In this section we analyze the computational complexity of our algorithms.

The page segmentation algorithm works directly on the DOM tree of the Web page in a top-down fashion and its complexity is $nlog(n)$, where n is the total number of nodes in the DOM tree. The group hierarchy inference phase iteratively goes through the path sequence in a bottom-up fashion until no more regular expressions are found. Assuming there are k nodes in the segment, the worst complexity of this phase is k^3. Since k is considerably smaller than the number of leaf-nodes in the Web page, this is reasonable for the Web. The promotion and labeling phases are linear in the number of nodes in the corresponding group.

8 Experimental Results

In our experiments, we used two types of data sets: template driven ones and non-template driven ones. Since we could not find publicly available annotated non-template driven, we prepared our own data set which is composed of faculty and course home pages of computer science departments. As the template driven data set, we selected TAP[2]. We will give the details of setup, the results and discussions for each experiment in the following subsections. Experimental data and evaluation results are also available online at `http://cips.eas.asu.edu/ontominer_files/eval.htm`.

[2] Home page is located at `http://tap.stanford.edu`

DomainName	P(C)	R(C)	F(C)	P(A)	R(A)	F(A)	P(V)	R(V)	F(V)
CIA	100	100	100	96	54	69	88	99	92
FAS Military	100	100	100	96	76	85	84	99	91
Great Buildings	100	100	100	95	61	74	88	99	93
Missile Threat	100	100	100	96	63	76	66	99	79
Roller Coster Database (RCDB)	100	100	100	78	52	46	84	91	88
Tower Records	100	100	100	75	69	72	62	55	58
WHO	100	100	100	100	94	97	85	100	92
Overall	100	100	100	91	67	74	79	92	85

Table 1. Experimental results with TAP data set using Semantic Partitioner algorithm. P(C), R(C), F(C) denote the precision, recall and F-measure of the concept annotation. Similar notation is used for attributes (A), values (V) and nones (N).

8.1 Setup

TAP Dataset: The TAP data set contains selected categories from TAP Knowledge Base 2 (TAP KB2) [12], including AirportCodes, CIA, FasMilitary, GreatBuildings, IMDB, MissileThreat, RCDB, TowerRecords and WHO. These categories alone comprise $9,068$ individual Web pages and these Web pages are attribute-rich. We provide experimental results for this data set with our algorithms and compare them against the relations obtained by TAP.

CSEDepts Dataset: The CSEDepts data set consists of 60 computer science department Web sites, comprising $7,793$ individual Web pages and $27,694$ total number of labels. The computer science department Web sites are *meta-data-driven*, i.e., they present similar meta-data information across different departments. To demonstrate the performance of our meta-tagging algorithms, we created a smaller data set containing randomly chosen 120 Web pages from each of the *faculty* and *course* categories. We provide experimental results for this data set with our algorithms.

8.2 Experiments with the TAP Data Set

The Table 1 show the experimental results for the TAP data set using semantic partitioning algorithm. The algorithm achieves 100% accuracy with tagging the labels with the concept label. Since the TAP data set contains only one concept per page, the algorithm is able to easily identify the label. However, the algorithm suffers from low recall with tagging as attribute labels because some of the attribute labels are single valued and there is no group associated with them, and they are labeled as values. However the algorithm is able to tag with the attribute labels correctly whenever it does, as the precision is above 91%. As expected, the recall and precision numbers for the value label tagging are exactly opposite for the same reasons that many attribute labels are labeled as values. But the results look very impressive for a completely automated system that is able to perform without building any wrappers for specific Web sites.

Domain	P(C)	R(C)	F(C)	P(A)	R(A)	F(A)	P(V)	R(V)	F(V)	P(N)	R(N)	F(N)
Courses	0	0	0	64	42	50	61	75	76	21	90	37
Faculty	50	0	4	51	44	47	65	80	71	34	31	32
Overall	25	0	2	57	43	49	70	78	73	27	61	35

Table 2. Results from CSEDepts Dataset. These results have been computed by manually evaluating the role assignments. P(C), R(C), F(C) denote the precision, recall and F-measure of the concept annotation. Similar notation is used for attributes (A), values (V) and nones (N).

8.3 Experiments with the CSEDepts Data Set

Recalling the motivating example, one can observe that labels in the faculty and course home pages of computer science departments are highly ambiguous. Many labels have different roles in different Web pages depending on the context.

Table 2 demonstrates that the initial annotations by the semantic partitioner algorithm achieves a reasonable overall accuracy around 60%. The F-measure for the concepts is very low because the number of concept labels in the domain are very few and they are ambiguously located along with the other attribute labels. However the algorithm is able to perform fairly accurately in annotating the attribute and value labels which is very vital because, these labels are the most frequent on the Web. The F-measure for value annotation is the highest among all role annotations, as the semantic partitioner is able to correctly identify the groups and thus identify the instances of the concept and attribute labels accurately.

These experimental results are obtained by comparing the data annotations of the algorithms to manually annotated data by eight human volunteers. The inter-human agreement on manual annotation was 87%, which indicates that the data annotations can be ambiguous and can be interpreted differently in various contexts. But our algorithm is able to perform well even in the presence of such ambiguity.

9 Conclusions & Future Work

In this paper, we have presented a system that can automatically gather and separate meta-data and their instances from various kinds of Web pages. The experimental results indicate that our algorithms were able to extract the meta-data and the instance information with high accuracy. In our future work, we identify the missing attribute labels to the values in the Web pages by identifying them using the contextual information. We plan to investigate the utility of our algorithms in extracting meta-data information for each group structure on the Web.

References

1. Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 1999.
2. Nickolas Kushmerick, Daniel S. Weld, and Robert B. Doorenbos. Wrapper induction for information extraction. In *Intl. Joint Conference on Artificial Intelligence (IJCAI)*, pages 729–737, 1997.
3. Naveen Ashish and Craig A. Knoblock. Semi-automatic wrapper generation for internet information sources. In *Conference on Cooperative Information Systems*, pages 160–169, 1997.
4. Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Web-scale information extraction in knowitall. In *Intl. World Wide Web Conf.*, 2004.
5. Fabio Ciravegna, Sam Chapman, Alexiei Dingli, and Yorick Wilks. Learning to harvest information for the semantic web. In *Proceedings of the 1st European Semantic Web Symposium*, Heraklion, Greece, 2004.
6. Stephen Dill, John A. Tomlin, Jason Y. Zien, Nadav Eiron, David Gibson, Daniel Gruhl, R. Guha, Anant Jhingran, Tapas Kanungo, Sridhar Rajagopalan, and Andrew 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. Guizhen Yang, Wenfang Tan, Saikat Mukherjee, I.V.Ramakrishnan, and Hasan Davulcu. On the power of semantic partitioning of web documents. In *Workshop on Information Integration on the Web*, Acapulco, Mexico, 2003.
8. 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.
9. Marti A. Hearst. Untangling text data mining. In *Association for Computational Linguistics*, 1999.
10. 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.
11. Arvind Arasu and Hector Garcia-Molina. Extracting structured data from web pages. In *ACM SIGMOD*, San Diego, USA, 2003.
12. R.V. Guha and Rob McCool. Tap: A semantic web toolkit. *Semantic Web Journal*, 2003.
13. Robert B. Doorenbos, Oren Etzioni, and Daniel S. Weld. A scalable comparison-shopping agent for the world-wide web. In W. Lewis Johnson and Barbara Hayes-Roth, editors, *Proceedings of the First International Conference on Autonomous Agents (Agents'97)*, pages 39–48, Marina del Rey, CA, USA, 1997. ACM Press.
14. Minos 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. Deng Cai, Shipeng Yu, Ji-Rong Wen, and Wei-Ying Ma. Vips: a vision-based page segmentation algorithm. Technical Report MSR-TR-2003-79, Microsoft Technical Report, 2003.
16. Soumen Chkrabarti. Integrating the document object model with hyperlinks for enhanced topic distillation and information extraction. In *10th International World Wide Web Conference*, 2001.