A Highly Adaptable Web Information Extractor Using Graph Data Model

Qi Guo, Lizhu Zhou, Zhiqiang Zhang, Jianhua Feng

Tsinghua University, Beijing, China
guoqi00@mails.tsinghua.edu.cn

Abstract. We present an approach to build highly adaptable extractor for collecting data from diverse Web sites. This approach uses Graph Model to represent content and structures as well as their various types of features. The generated graph is accompanied by a script in a special language called GQML containing the extraction rules. The running of the script transforms the graph into a specified format such as XML file that stores data from various Web sites in a uniform format. The experimental results show the presented approach is both effective and efficient.

1. Introduction

Web site information sharing and exchanging between different applications are difficult. A popular approach to address this problem is to write wrappers [1] to encapsulate heterogeneity in accessing diverse data sources. Generally, each data source requires an individual wrapper which needs manual programming. Instead of using this one-for-one method, we introduce an approach that develops only one wrapper highly adaptable to diverse Web data sources.

The basic idea of the approach is providing a script interface for user to present the required data schema and providing a powerful, easy-to-use and flexible facility to describe the extract rules. The information extraction process in our approach is divided into four steps. First, reconstruct original HTML string into a directed labelled graph. Second, supply the graph by adding nodes and labels which represent various features of the contents or formats. Third, transform the graph into final form containing the extracted objects. Finally, export the results into XML files.

There are many approaches that are comparable to ours. One is to restructure the original HTML into an internal representation, and provide a specific wrapper generation language, such as TSIMMIS[2], WebOQL[3], etc. The other way is to provide GUI or interactive tools to help user build wrappers easier. For instances, XWrap[4] and Lixto[5] adapt this approach. All of these systems work for limited data sources. Ontology based technology are also used in this area, for instance, the Data Extraction Group[6] has built a domain ontology to describe data of interest. But the construction of ontology needs careful work by domain expert, and the features of hypertext are not adequately cared.

2. Graph Construction

A directed labelled graph consists of a finite set of labelled nodes and a finite set of labelled directed edges. Edge between node x, y is denoted as *a*(x, y), with the label of *a*. A graph can be represented as {a1(x1, y1),a2(x2,y2),….., an(xn, yn)}. In our approach, each node represents either a text, or a region matching specific features of a Web page, while each edge represents the relation between the two nodes.

A Web site HTML page can be mapped into a graph in following way, first separate the original HTML string into finite tokens delimited by inter-punctuations and HTML tags; then, map every token into a node, and add labelled edges between adjoined tokens or continuously blocks; at last, annotate regions that match predefined feature annotation rules.

3. Graph Rewriting Rule

Graph rewriting technique has been widely used in varies applications, especially in database area [7]. A graph rewriting rule is applied to a host graph to replace one subgraph by another. We define the original graph as the graph mapped by HTML file, and the final state of transformed graph as extracted object. The extractor generation problem is actually the generation of the graph rewriting rules. To facilitate this rewriting, we introduce a graph query and manipulate language, called *GQML*.

In GQML, the simplest query pattern is a constraint to a single node or edge, for instance, find nodes whose value equals "A", or find pairs of nodes which jointed with edge labelled "B". In addition, GQML provides a series of rules to represent complex query patterns [8].

The manipulate operations are often composed with graph query. Assume Query Q(a1, a2, … an) is a query, M(a1, a2, … an) is a manipulate function, an complete graph rewriting rule is Q.M (a1, a2, …,an). Figure 1 shows this rewriting process.

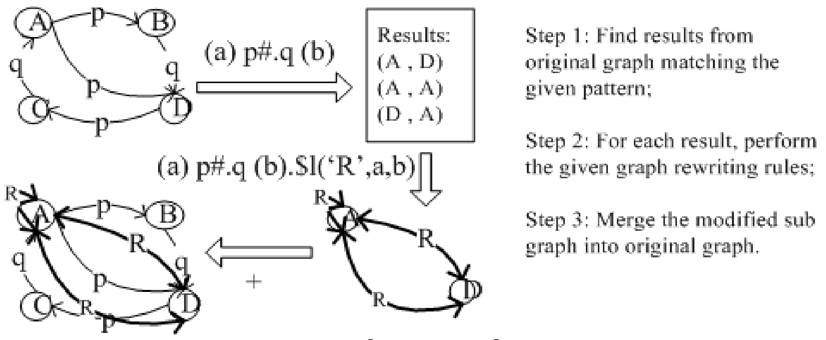
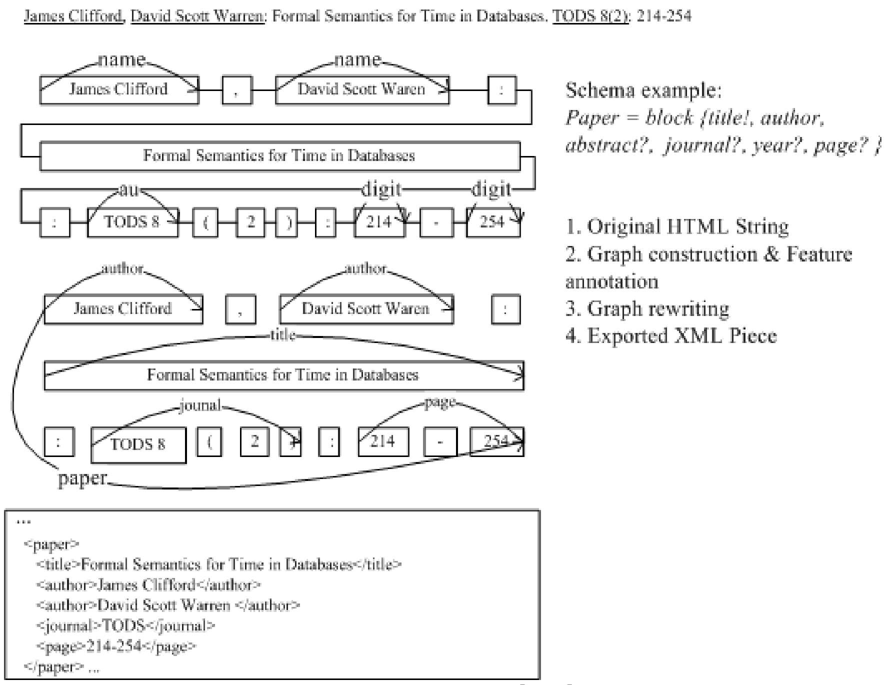

(a) p#.q (b)
Results:
(A , D)
(A , A)
(D , A)
(a) p#.q (b).$l('R',a,b)
+
Step 1: Find results from original graph matching the given pattern;
Step 2: For each result, perform the given graph rewriting rules;
Step 3: Merge the modified sub graph into original graph.

Figure 1: Graph Query and Rewriting

After a sequence of graph rewriting, the original graph is transformed into a new graph that contains the required data. A predefined schema is used to tell how these data should be organized. According to this schema, the data are extracted and exported into an XML file. Figure 2 gives an example.


James Clifford, David Scott Warren: Formal Semantics for Time in Databases. TODS 8(2): 214-254
name
James Clifford
,
name
David Scott Waren
:
Formal Semantics for Time in Databases
au
TODS 8
(
2
)
:
digit
214
-
digit
254
author
James Clifford
author
David Scott Waren
title
Formal Semantics for Time in Databases
jounal
TODS 8
page
214
254
paper
Schema example:
Paper = block {title!, author, abstract?, journal?, year?, page? }
1. Original HTML String
2. Graph construction & Feature annotation
3. Graph rewriting
4. Exported XML Piece

```
...
<paper>
  <title>Formal Semantics for Time in Databases</title>
  <author>James Clifford</author>
  <author>David Scott Warren </author>
  <journal>TODS</journal>
  <page>214-254</page>
</paper> ...
```

Figure 2: Outputted xml

4. Feature Framework

Features play an important role in identifying Web page data. GQML supports four classes of features: syntax-feature, visual-feature, lexical-feature and boundary-feature. Besides, users can customize new features for specific application by a feature-extension interface. The interface provides two extension methods: single-token feature and multi-token feature. Single token feature covers only one token and represent by regular expression, while the multi token feature covers a continuous series of tokens. The detail is discussed in [8].

5. Implementation and Experiments

Using GQML, we have developed an application for collecting research papers of computer science area from some famous Web sites and 100 researcher's home pages. In this application, papers collected from these sites quite differ in their presentation format. But they are all treated by one wrapper with 7 graph rewriting rule and 1 schema [8]. The experiment results are listed in Table 1.

Data source	**CiteSeer**	**DBLP**	**Home Pages**
Number of Pages	1500	1000	100
Number of Objects	1500	3415	620
Exported/Corrected Objects	1319/981	2910/2475	715/430
Precision	74.3%	85.1%	60.3%
Recall Rate	65.4%	72.4%	49.4%
Process Time	31 s	17s	2.4s

Table 1: Experiment result

6. Conclusions

In this paper, we describe an approach to build highly adaptable wrapper. Our major contribution is the use of graph data modal to represent document's content, structure and feature, and to represent the extraction process as well. We also introduce a language called GQML to represent the graph rewriting rules. The approach is supported by an environment in building domain related web applications that require extracting data from various data source. Experiment shows that our method is effective and efficient.

References

1. N. Ashish, C. Knoblock. Wrapper generation for semistructured internet sources. Proceedings of the Worshop on Management of Semi-structured Data, 1997.
2. J. Hammer, J.Mchugh, H. Garcia Molina. Semistructured Data: The TSIMMIS Experience. In proceedings of the First East-European Symposium on Advances in Databases and Information Systems (ADBIS'97) pp. 1-8.
3. G. Arocena, A. Mendelzon. WEBOQL: Restructuring Documents, Databases, and Webs. In proceedings of the 14th IEEE International Conference on Data Engineering. p24-33.
4. L. Liu, C. Pu, W. Han. XWrap – An XML-enabled Wrapper Construction System for Web Information Sources, Proceedings of the 16th International Conference on Data Engineering (ICDE'2000)
5. R. Baumgartner, S. Flesca, G. Gottlob. Visual Web Information Extraction with Lixto, Paper for the 27th International Conference on Very Large Data Bases (VLDB 2001)
6. D.W. Embley, D. M. Campbell, el. Conceptual-Model-Based Data Extraction from Multiple-Record Web Pages. Data and Knowledge Engineering 31, 3(1999), 227-251
7. M. Main, G. Rozenberg. Edge-Label controlled Graph Grammars. *Journal of Computer and System Sciences*,Vol. 40, 1990, pp. 188-228.
8. Q. Guo, H. Guo, J. Sun, Z. Zhang, and L. Zhou. Technique Report on SESQ. http://dbgroup.cs.tsinghua.edu.cn/sesq