Semi-structured Data Extraction and Schema Knowledge Mining

Chen Enhong Wang Xufa
Department of Computer Science
University of Science and Technology of China
Hefei Anhui 230027 P.R.China

Abstract

It is well known that World Wide Web has become a huge information resource. Therefore, it is very important for us to utilize this kind of information effectively. This paper proposes a semi-structured data extraction method to get the useful information embedded in a group of relevant web pages, and store it with OEM(Object Exchange Model). Then, we adopt data mining method to discover schema knowledge implicit in the semi-structured data. This knowledge can make users understand the information structure on the web more deeply and thourouly. At the same time, it can also provide a kind of effective schema for the querying of web information.

1. Introduction

1.1 Overview

It is well known that WWW has become a huge information resource. Therefore, it is very important to effectively utilize this kind of information[1]. However, the information on WWW can not be queried and manipulated in a general way. Vast amount of information is stored in a static HTML format and can only be viewed with browsers. Although some sites may provide search engines, the queries are performed through keyword match operations, and query results are still in HTML format. The information still needs to be viewed on the corresponding web sites through browsers. Users are difficult to obtain the information structure and schema information of the whole web site.

The paper implements a data extraction method, which extracts useful information from a group of relevant web pages. In fact, this kind of information does not have any predefined structure, and it is also called semi-structured data[2]. It appears in a wide range of applications, such as digital libraries, on-line documentations, electronic commerce[3]. Because it is hard to be represented with traditional relational model, the paper adopts OEM(Object Exchange Model) [4] to represent it. After we have obtained enough data from WWW, we then use data mining[5-7] method to mine schema knowledge from the data. Unfortunately, most of existing methods focus on the knowledge discovery from relational data, we must design a new method to deal with the hierarchy and irregularity of the semi-structured data. In different data, the same attribute might have different number of values, or even have none. Therefore, we design an algorithm based on frequent itemset discovery of association rule mining method. The schema knowledge mined can provide users with an overall understanding of the information on the web site, and it also can provide an effective schema for the query on the web.

1.2 Outline of the Paper

In the following section, we will describe semi-structured representation model with an example. In section 3, the detail of the implementation of extraction will be given. In section 4, we will introduce the method of mine schema knowledge from semi-structured data. In the final section, some conclusion will be given.

2. Semi-structured Data Representation Model

In this section, we will introduce Object Exchange Model (OEM) for representing Semi-structured data by examples. In OEM, each object contains an object identifier and a value. A value may be atomic or complex. Atomic values may be integers, real, strings, images, program. A complex OEM value is a collection of 0 or more OEM sub-objects, each linked to the parent via a descriptive textual label.

Fig.2 is the directional graph representation of OEM model for some information extracted from *http://us.imdb.com/Title?Above+and+Beyond+(1952)* presented in Fig.1. In the figure, *WonNom* represents *Won* or *Nominated.*

```
Root complex{
  Film complex{
    Name string “Above+and+Beyond”
    Year string “1952”
   Awards url http://More?tawards+Above+and+Beyond
+(1952)
    Genre complex{
      Keyword string “Action”
      Keyword string “Biographical”
      Keyword string “Atomic-Weapons”
    }
    director url http://Name?Melvin,+Frank
    }
  http:// Name? Melvin,+Frank   complex{
    Name string “Melvin Frank”
    Awards url http:// More?tawards+Melvin,+Frank
  }
  http://More?tawardsAbove+and+Beyond+(1952)
Complex{
    WonNon string “Oscar”
  }
  http:// More?tawards+Melvin,+Frank Complex{
    WonNon string “Golden Globe”
  }
…
}
```

Fig.1 An example of Semi-Structured Data from http://us.imdb.com/Title***?Above+and+Beyond+ (1952)***

3. The Implementation of Semi-structured Data Extraction

In our implementation, users should provide an initial http address to Semi-Structured Data Extractor. Then Extractor start to get the needed HTML file from corresponding remote web server, extract the useful data based on the specification file directing the extraction, and store it in OEM model. If some useful hyperlinks are detected during the process of extraction, then these hyperlinks will be inserted in a queue so that the system adopts breadth-first strategy to get HTML files and extract data. After the extraction task has been finished, the semi-structured data can be used for schema knowledge discovery.

Semi-structured data has no fixed schema, and the same attributes have different number of values or even have no values in different but similarly structured web pages. Therefore, it makes extraction task very difficult. Considering that web pages are stored in HTML format, we design a kind of specification file for every class of

similarly structured web pages. The specification file is designed for extracting values for interesting attributes, and then adding labels. Fig.3 is an example of the file used to extract information on film pages on web site *http://us.imdb.com*. Line 2 denotes the match pattern before the information (represented by star) needed to be extracted. Line 3 denotes the label to be added. Line 4 denotes the number of values to be extracted. When a hyperlink is extracted, we must tell the program which class the hyper-linked page belongs to. For example, Line

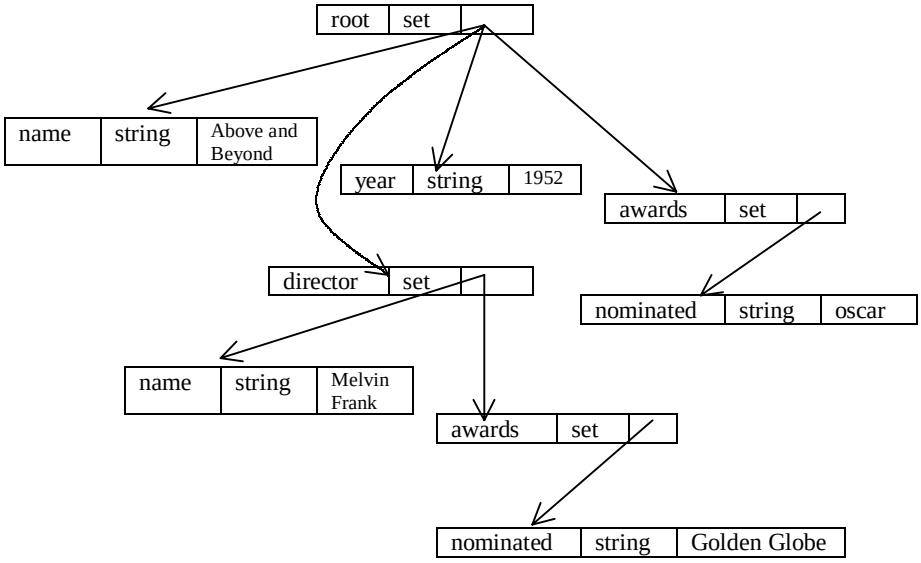

Fig.2 The directional graph representation of OEM model.

10 denotes that a hyperlink with the prefix HREF=”/More?tawards+ links to page of Class 1.

```
1 [
2 Extract          <TITLE>*
3 Add label:       Name
4 Num of Values:   1
5 ]
   ……
6 [
7 Extract:         HREF=”/More?tawards+*
8 Add label:       Award
9 Num of Values:   1
10 Page type:      1
11 ]
   ……
```

Fig.3 An Example of Extraction Specification File

Algorithm 1 is for extracting semi-structured data from WWW and storing it in OEM model. This algorithm first gets an HTML file Doc from a web site. Then it extracts the needed information guided by the corresponding specification file. The subroutine P(*S, Tag(f)*) performs a particular data extraction task, in which there are two cases needed to deal with. The first case is that the information *V* followed by Cur_tag needs to be extracted. In such case, if *V* is atomic, then we add <*label, V*> to OEM database, in which label is obtained from the specification file. If *V* is a hyperlink pointed to another page, then we append *V* and

the specification file number for extracting corresponding web pages to the tail of queue *Q*. In the second case, the algorithm has detected *Cur_tag*. This means that the contents following *Cur_tag* in the file have no more values for the current attribute. We should extract values for other attributes.

Algorithm 1: extract_info( )
Input: Q: Queue to store the http address;
Output: Semi-structured Data Represented by OEM;
{
Match ← True;
While (*Q*<>*empty*) do {
addr ← first entry in Q;
get an HTML document Doc(*addr*) from remote web server;
read the corresponding tag file Tag(Doc(addr));
repeat {
if (*Match* == True or *Cur_tag* == NULL) then
S ← the starting position of next string in Doc(*addr*);
Cur_tag ←Current tag in Tag(Doc(*addr*));
if (*Cur_tag* is the prefix pointed by *S*) then
Match = True;
P(S, Tag(Doc(*addr*)));
else advance the pointer in Tag(Doc(*addr*));
Match = False;
endif
}Until *EOF*(Doc(*addr*)) or *EOF*(Tag(Doc(*addr*)))
} end while
}

4. Schema Knowledge Discovery for Semi-Structured Data

After having extracted enough semi-structured data, we can mine knowledge for many purposes. In the following, we will give some details for mining schema knowledge from semi-structured data.

4.1 Some Definitions

The algorithm is based on the discovery method of frequent itemsets of association rule algorithm. However, the traditional methods are only suitable for flat structured data. Therefore, we must make great modification about it. First we need to define Transaction in a different way.

Definition 1 **Extension**: Given an object *O*, if *O* has n outgoing edges, with li labeled on each edge and ending in object O_i. (1) $Ext(O) = \{<l_1, O_1>, \ldots, <l_n, O_n>\}$ is a direct extension of *O*. (2) If $Ext(O_{i_j})$ is an extension of object O_{i_j}, $\{<l_{i_1}, Ext(O_{i_1})>, \ldots, <l_{i_m}, Ext(O_{i_m})>\}$ is a generalized extension of object *O*, i_j {1, 2, …, n}, where $m<=n$.

Definition 2 **Transaction**: Given a complex object T whose direct extension is $Ext(T) = \{<l_1, T_1>, \ldots, <l_n, T_n>\}$. If there is no any object which includes *Ext(T)* as an element in its extension , then we call $Ext(T) = \{<l_1, T_1>, \ldots, <l_n, T_n>\}$ a transaction.

From the above definition, if transaction T_1 has n outgoing edges, with l_i labeled on the *i-th* edge ending in object O_i, then we can denote T_1 by first level extension $Ext(T_1) = \{<l_1, O_1>, \ldots, <l_n, O_n>\}$, which means that T_1 is extended by its sub-objects with labels. In fact, such kind of first level extensions of *T* serve as the record of the transaction database. The lower level extension will be used in the process of mining.

Let's see an example shown in Fig.4, it represents a transaction *T* ={<*Name, Above and Beyond*>, <*Year, 1952*>, <*Genre, Keyset*>, ..., <*Director, Melvin Frank*>}, where Keyset is also a complex object whose extension is *Ext(Keyset)* ={<*Keyword, action*>, <*Keyword, biography*>}.

As mentioned above, in the representation of semi-structured data, label expresses semantic information. Therefore, we must explicitly include labels in transactions. For example, in <*Name, Above and Beyond*>, *Name* is label, *Above and Beyond* is an atomic object.

Definition 3 **Frequent *k*-schema**: A *k*-schema is a

generalized extension with *k* atomic *objects*, i.e. each object has no extension. A *k*-schema whose support is greater than the user-defined threshold is called frequent *k*-schema.

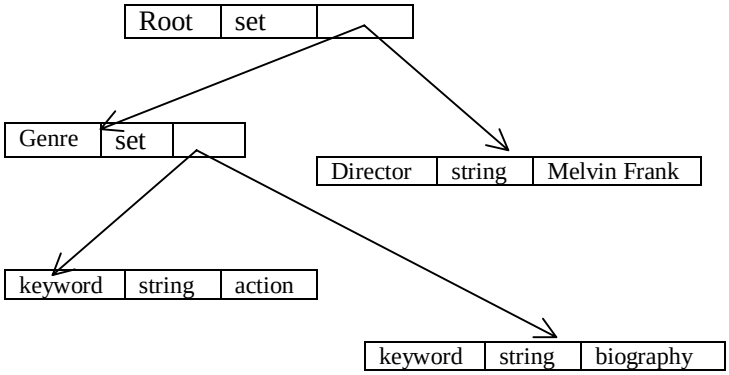

Fig.4 The Directional Graph Representation

As shown in Fig.4, its generalized extension {<*Genre*, {<*Keyword*, *Action*>, <*Keyword*, *Biographical*>}>, <*Director*, *Melvin Frank*>} is a *3*-schema, for this schema has 3 atomic objects, which are *Action* with label *Keyword* <*Keyword*, *Action*>, *Biographical* with label *Keyword* <*Keyword*, *Biographical*>, and *Melvin Frank* with label *Director*< *Director*, *Melvin Frank*>. Actually, we ignore the internal objects in the *k*-schema.

4.2 Schema Discovery Algorithms

From the definition given above, we see that the first level nodes in directional graph are treated as transactions. The lower level nodes are used in later mining process to find frequent schema. In fact, this is the schema knowledge implicit in semi-structured data. The overall structure of our algorithm is similar to that given in [5] which discover large schema from transactions. However, as pointed above, for transactions here are in different form, that is, changing from flat relational data to hierarchical semi-structured data. The generation of candidate frequent k-schema is in a different way. Furthermore, we can obtain two types of schema knowledge depending on whether we include object values or not. The first is structure schema, which contains label information and ignores object values. This schema is actually the structure implicit in all transactions. The other is object association schema, which contain both label and object information, and gives us the information about which objects are often appears together.

To facilitate the implementation of the algorithm for mining schema, we have introduced a hash table. When trying to get an extension of an object, we index on the object and can easily find all its sub-objects.

K- schema can be viewed as a tree structure. We call it a descending tree. The tree can be represented as sequence $PT_1 ... PT_k$, in which PT_i is the descending path for the *i-th label* sequence that ending in object O_i in the schema, i.e., $\{l_1, \ldots, < l_n, id_n> \}$. Fequent *1*-schema can be discovered through finding descending path whose support is greater than a preset threshold. Furthermore, no internal node objects are included in schema. This makes all descending path with label sequences l_1, l_2, ..., l_n which terminate in the same node be the same 1- schema. After finishing the calculation of F_1, we can calculate all F_k, where $k \geqslant 2$, through combining $PT_1 ... PT_{k-1}$ and $PT_1 ... PT_{k-2}\ PT_k$ to

construct PT_1... PT_{k-2} PT_{k-1} PT_k.

For example, given transactions $T1$ and $T2$ shown in fig.5, we can get two 1- schema PT_1 = {l_1, <l_3, O_4>}, PT_2 = { l_1, l_4, <l_5, O_7>} supported by transactions $T1$ and $T2$. Based on PT_1 and PT_2, we can get a 2-schema PT_1PT_2 = {l_1, {{<l_3,O_4>}, {l_4, <l_5, O_7>}}} supported by $T1$ and $T2$.

Algorithm 2 is for generating frequent k-schemas from transaction database containing semi-structured data represented in OEM model. Considering that two *1*-schema can be combined many possible *2*-schema, we give each label an order. If an object has more than 1 outgoing edges sharing a common label, then we associate each label a number, 1 for the first label, 2 for the second label, etc. Otherwise, we simply associate each label with 1. For example, in Algorithm 2, $l_m(i_m)$ in descending path PTi_j = {$l_1(i_1)$, $l_2(i_2)$, …, < l_k (i_k), O_k>} denotes that label l_m is the i_m-th occurrence for the outgoing edges of object O_m. In this way, we can eliminate many impossible candidate k-schema in the following process.

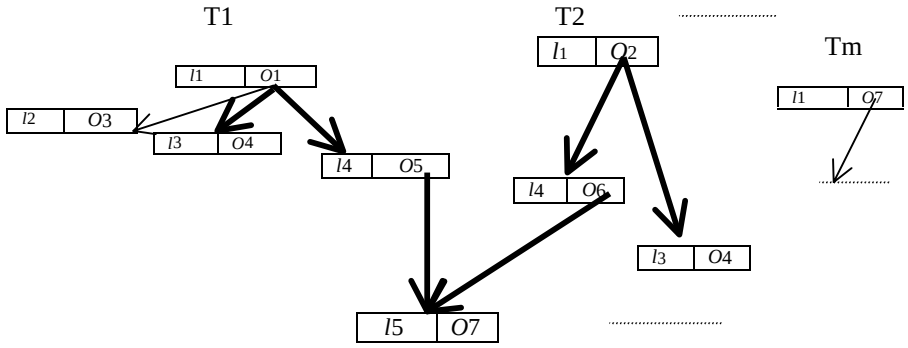

Fig.5 The 2-schema supported by transaction T1 and T2 are
those with thick black arrows and the connected nodes

Algorithm 2: Generating *k*-schema()
Input: transaction database;
Output: frequent *k*-schema;
{
for each transaction *Ext(T)* do
for each descending path PTi_j = {$l_1(i_1)$, $l_2(i_2)$, …, < $l_k(i_k)$, O_k>}do
if PT_i is traversed for the first time in *Ext(T)* then
++support(PT_i);
for each *PT* do{
if support(*PT*) > *minsup* then{
add *PT* = {$l_1(i_1)$, $l_2(i_2)$, …, < $l_k(i_k)$, O_k>} to FS_1;
store all the order {i_1, i_2, …, i_k}associated with *PT*;
}
for (*k*=2; FS_i <> 0; *k*++) do{
CFS_k = generate_candidate_schema(FS_{k-1});
for each transaction *EXT(T)* do
For each candidate *k*-schema c in CFS_k do{
if sub_structure(*c*, *EXT*(*T*)) then
++support(*c*);
for each candidate *k*-schema *c* in CFS_k do
if support(*c*) > *minsup* then
add *c* to FS_k;
};
output $FS_1 \cup \ldots \cup FS_{k-1}$;
}

4.3 An Example

As an example, we have extracted semi-structured

data from more than 18k film web pages on site *http://us.imdb.com*. It involves more than 100k HTML files for many castors, writers, directors and editors appearing on film pages have hyperlinks to their own pages on the same web site. The size of the extracted semi-structured database reaches 20M. Fig.6 is some content of HTML file for the web page http://us.imdb.com/Pawards? Lucas,+George. The page provides some information of awards received by George Lucas. The darkened are the information we have extracted.

<HTML>
<HEAD>
<BASE TARGET="_top">
<TITLE>Awards information for George Lucas </TITLE>……<TR><TD ROWSPAN="2" ALIGN="CENTER"VALIGN="CENTER">***1978***</TD><TD ROWSPAN="2" ALIGN="CENTER" VALIGN="CENTER"><B CLASS="silver"> ***Nominated*** </B></TD><TD ROWSPAN="2" ALIGN="CENTER" VALIGN="CENTER"> **Oscar**</TD><TD VALIGN ="CENTER">***Best Director***
<B CLASS="smallkey" >for: </B><A HREF="/More?tawards+Star+Wars+ (1977)">Star Wars (1977)</A>
……

Fig.6 Part of HTML file content for web page http://us.imdb.com/Pawards?Lucas,+George

Fig.7 is an example of the part of structure association schema which ignores object values. This schema states the information contained on the film and its hyper-linked pages. Information for directors includes Name, Birthday, Birthplace, Name and Occupation for their spouses, etc.

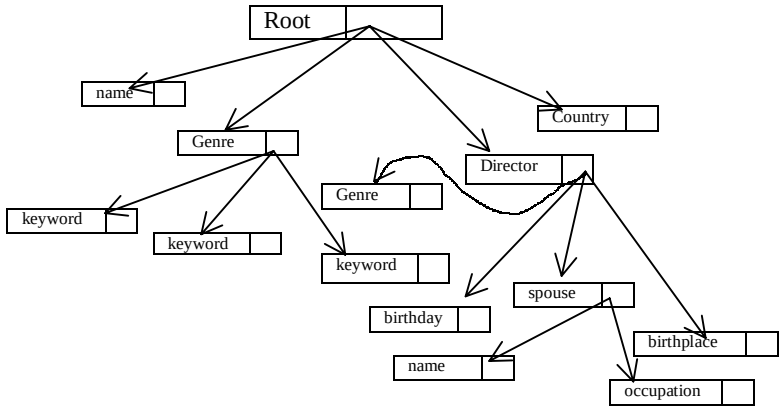

Fig7. An example of the schema mined from http://us.imdb.com

5. Conclusion

The paper focuses on the semi-structured data extraction and schema knowledge mining. With the rapid growth of WWW, the semi-structured data will be richer and richer, and it will certainly attract more and more researchers to study it. In the future, we will further the work on two directions. The first is to introduce machine learning method to the recognition of tag information in extraction. We will also study the clustering method in semi-structured data knowledge discovery.

References

[1] T. Berners-Lee, R. Cailliau, A. Luotonen, H. F. Nielsen and A. Secret. The World Wide Web. Comm. Of ACM, 37(8):76-82, 1994.

[2] D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, and J. Widom. Querying Semistructured Heterogeneous Information. In Deductive and Object-Oriented Databases(DOOD):319-344,
Singapore, December 1995.

[3] S. Abiteboul. Querying Semi-structured Data. In Proceedings of International Conference on Database Theory, 1997.

[4] S. Abiteboul, D. Quass, and J. McHugh, J. Widom, J.L. Wiener. The Lorel Query Language for Semi-structured Data. Journal of Digital Libraries, 1997.

[5] R.Agrawal, T.Imielinski, and A. Swami. Mining Association Rules between Sets of Items in Large Databases. In SIGMOD:207-216,1993.

[6] K. Wang, H.Liu. Discovering Typical Structure of Documents: A Road Map Approach. In ACM SIGIR Conf. On Research and Development in Information Retrieval. Aug. 1998.

[7] Q.Wang, E.Chen. Researches on Some Problems and Applications of KDD. Computer Science, 24(5):73-77, 1997.