Journal of Data Mining and Knowledge Discovery
ISSN: 2229–6662 & ISSN: 2229–6670, Volume 3, Issue 2, 2012, pp.-69-73.
Available online at http://www. bioinfo. in/contents. php?id=42

DEEP WEB DATA EXTRACTION USING WEB-PROGRAMMING-LANGUAGE-INDEPENDENT APPROACH

PALEKAR V.R.*, ALI M.S. AND MEGHE R.

Department of Information Technology, Institute of Technology & Research, Badnera, MS, India.
*Corresponding Author: Email- vikaspalekar@gmail.com

Received: March 15, 2012; Accepted: April 12, 2012

Abstract- The problem of Web data extraction has received a lot of attention in recent years and most of the proposed solutions are based on analyzing the HTML source code or the tag trees of the Web pages. These solutions have the following main limitations: First, they are Web-page-programming-language-dependent, or more precisely, HTML-dependent. As the popular two-dimensional media, the contents on Web pages are always displayed regularly for users to browse, this motivates us to develop new extraction technique. This approach primarily utilizes the visual features on the deep Web pages to implement deep Web data extraction, including data record extraction and data item extraction.

Keywords- Web Data Extraction, Data Iitem Extraction, Web Mining, Web-Programming-Language-Independent.

Citation: Palekar V.R., Ali M.S., Meghe R. (2012) Deep Web Data Extraction using Web-Programming-Language-Independent Approach. Journal of Data Mining and Knowledge Discovery, ISSN: 2229–6662 & ISSN: 2229–6670, Volume 3, Issue 2, pp.-69-73.

Introduction

World Wide Web has more and more online Web databases which can be searched through their Web query interfaces. The number of Web databases has reached 25 million according to a recent survey [5]. All the Web databases make up the deep Web (hidden Web or invisible Web). Often the retrieved information (query results) is enwrapped in Web pages in the form of data records. These special Web pages are generated dynamically and are hard to index by traditional crawler based search engines, such as Google and Yahoo.

The problem of Web data extraction has received a lot of attention in recent years and most of the proposed solutions are based on analyzing the HTML source code or the tag trees of the Web pages. These solutions have the following main limitations: First, they are Web-page-programming-language-dependent, or more precisely, HTML-dependent. As most Web pages are written in HTML, it is not surprising that all previous solutions are based on analyzing the HTML source code of Web pages. However, HTML itself is still evolving (from version 2.0 to the current version 4.01 and version 5.0) and when new versions or new tags are introduced, the previous works will have to be amended repeatedly to adapt to new versions or new tags. Furthermore, HTML is no longer the exclusive Web page programming language and other languages have been introduced, such as XHTML and XML (combined with XSLT and CSS). The previous solutions now face the following dilemma: should they be significantly revised or even ab andoned? Or should other approaches be proposed to accommodate the new languages? Second, they are incapable of h andling the ever-increasing complexity of HTML source code of Web pages. Most previous works have not considered the scripts, such as JavaScript and CSS, in the HTML files. In order to make Web pages vivid and colorful, Web page designers are using more and more complex JavaScript and CSS.

Motivations

As the popular two-dimensional media, the contents on Web pages are always displayed regularly for users to browse. This motivates us to seek a different way for deep Web data extraction to overcome the limitations of previous works by utilizing some interesting common visual features on the deep Web pages [1].

Literature Survey

A number of approaches have been reported in the literature for extracting information from Web pages. Some of the approaches are as:

Manual Approaches

The earliest approaches are the manual approaches in which languages were designed to assist programmer in constructing wrappers to identify and extract all the desired data items/fields. Some of the best known tools that adopt manual approaches are Minerva [3], TSIMMIS [4] and Web-OQL [2]. Obviously, they have low efficiency and are not scalable.

Semiautomatic Approaches

Semiautomatic techniques can be classified into sequence based and tree-based. The former, such as WIEN [6], Soft-Mealy [7] and represents documents as sequences of tokens or characters and generates delimiter based extraction rules through a set of training examples.

The latter, such as XWrap [7], parses the document into a hierarchical tree (DOM tree), based on which they perform the extraction process. These approaches require manual efforts, for example, labelling some sample pages, which is labour-intensive and time-consuming.

Automatic Approaches

In order to improve the efficiency and reduce manual efforts, most recent researches focus on automatic approaches instead of manual or semiautomatic ones. Some representative automatic approaches are Omini [2]. These approaches perform only data record extraction but not data item extraction. In this approach they identify patterns and perform extraction for each Web page directly without using previously derived extraction rules.

Proposed System Design and Implementation

This approach will be implemented in four steps:

1) Given a sample deep Web page from a Web database, obtain its visual representation and transform it into a Visual Block tree.
2) Extract data records from the Visual Block tree.
3) Partition extracted data records into data items and align the data items of the same semantic together.
4) Generate visual wrappers (a set of visual extraction rules) for the Web database based on sample deep Web pages

Visual Block Tree and Visual Feature

Before the main techniques of this proposed project, we describe the basic concepts and visual features that our approach needs.

Visual Information of Web Pages

The information on Web pages consists of both texts and images (static pictures, flash, video, etc.). The visual information of Web pages used in this paper includes mostly information related to Web page layout (location and size) and font.

Web Page Layout

A coordinate system can be built for every Web page. The origin locates at the top left corner of the Web page. The X-axis is horizontal left-right and the Y-axis is vertical top-down. Suppose each text/image is contained in a minimum bounding rectangle with sides parallel to the axes. Then, a text/image can have an exact coordinate (x, y) on the Web page. Here, x refers to the horizontal distance between the origin and the left side of its corresponding rectangle, while y refers to the vertical distance between the origin and the upper side of its corresponding box. The size of a text/image is its height and width. The coordinates and sizes of texts/images on the Web page make up the Web page layout.

Font

The fonts of the texts on a Web page are also very useful visual information, which are determined by many attributes as shown in Table 1. Two fonts are considered to be the same only if they have the same value under each attribute.

Table 1- Font Attributes and Examples

Font Factor	Example	Font Factor	Example
Size	A(10pt)	Underline	A
Face	A (Bell MT)	Italic	A
Color	A (red)	weight	A
strikethrough	A	frame	A

Deep Web Page Representation

The visual information of Web pages, which has been introduced above, can be obtained through the programming interface provided by Web browsers (i.e., IE). In this paper, we employ the VIPS algorithm [4] to transform a deep Web page into a Visual Block tree and extract the visual information. A Visual Block tree is actually a segmentation of a Web page. The root block represents the whole page and each block in the tree corresponds to a rectangular region on the Web page. The leaf blocks are the blocks that cannot be segmented further and they represent the minimum semantic units, such as continuous texts or images. Fig. 2a shows a popular presentation structure of deep Web pages and Fig. 2b gives its corresponding Visual Block tree. An actual Visual Block tree of a deep Web page may contain hundreds even thous ands of blocks.

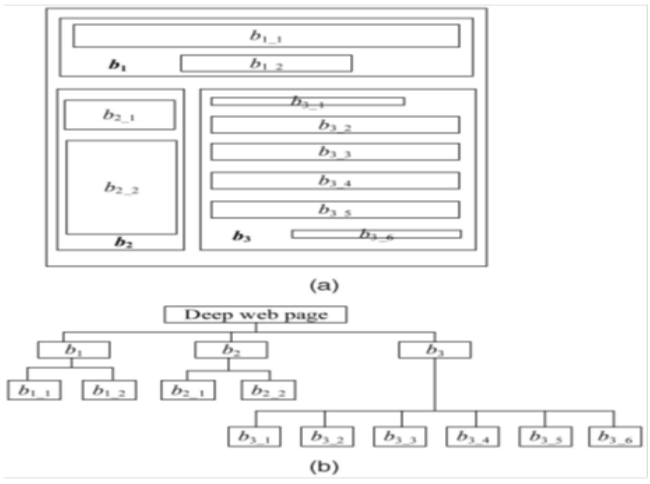

Fig. 1- (a) The presentation structure and **(b)** its Visual Block tree

Visual Block tree has three interesting properties. First, block a contains block b if a is an ancestor of b. Second, a and b do not overlap if they do not satisfy property one. Third, the blocks with the same parent are arranged in the tree according to the order of the corresponding nodes appearing on the page. These three properties are illustrated by the example in Fig. 2. The formal representations for internal blocks and leaf blocks in our approach are given below. Each internal block a is represented as a = (CS; P; S; FS; IS), where CS is the set containing its child blocks (note that the order of blocks is also kept), P is the position of a (its coordinates on the Web page), S is its size (height and width), FS is the set of the fonts appearing in a and IS is the number of images in a. Each leaf block b is represented as b = (P; S; F; L; I;C), where the meanings of P and S are the same as those of an inner block, F is the font it uses, L denotes whether it is a hyperlink text, I denotes whether it is an image and C is its content if it is a text.

Visual Features of Deep Web Pages

Web pages are used to publish information to users, similar to other kinds of media, such as newspaper and TV. The designers often associate different types of information with distinct visual characteristics (such as font, position, etc.) to make the information on Web pages easy to underst and. As a result, visual features are important for identifying special information on Web pages. Deep Web pages are special Web pages that contain data records retrieved from Web databases and we hypothesize that there are some distinct visual features for data records and data items. Our observation based on a large number of deep Web pages is consistent with this hypothesis. We describe the main visual features in this section and show the statistics about the accuracy of these features at the end of this Section 3.3.

Position features (PFs)

These features indicate the location of the data region on a deep Web page.

PF1: Data regions are always centered horizontally.

PF2: The size of the data region is usually large relative to the area size of the whole page.

Layout features (LFs)

These features indicate how the data records in the data region are typically arranged.

LF1: The data records are usually aligned flush left in the data region.

LF2: All data records are adjoining.

LF3: Adjoining data records do not overlap and the space between any two adjoining records is the same.

Appearance features (AFs)

These features capture the visual features within data records.

AF1: Data records are very similar in their appearances and the similarity includes the sizes of the images they contain and the fonts they use.

AF2: The data items of the same semantic in different data records have similar presentations with respect to position, size (image data item) and font (text data item).

AF3: The neighbouring text data items of different semantics often (not always) use distinguishable fonts.

Content feature (CF)

These features hint the regularity of the contents in data records.

CF1: The first data item in each data record is always of a m andatory type.

CF2: The presentation of data items in data records follows a fixed order.

CF3: There are often some fixed static texts in data records, which are not from the underlying Web database.

This deep Web data extraction solution is developed mainly based on the above four types of visual features. PF is used to locate the region containing all the data records on a deep Web page; LF and AF are combined together to extract the data records and data items.

Special Supplementary Information

Several types of simple non-visual information are also used in our approach in this paper. They are same text, frequent symbol and data type, as explained in Table 4. Obviously, the above information is very useful to determine whether the data items in different data records from the same Web database belong to the same semantic. The above information can be captured easily from the Web pages using some simple heuristic rules without the need to analyze the HTML source code or the tag trees of the Web pages. Furthermore, they are specific language (i.e., English, French, etc.) independent.

Table 2- Non-visual Information Used

Special complementary information	Remark
Same text	Given two texts, we can determine whether or not they are same
Frequent symbol	Given the deep web pages of a web database, if some symbols/words (e.g. ISBN, $) appear in all the data items of an attribute, they are called frequent symbols.
Data type	They are predefined, including image, text, number, date, price, email, etc.

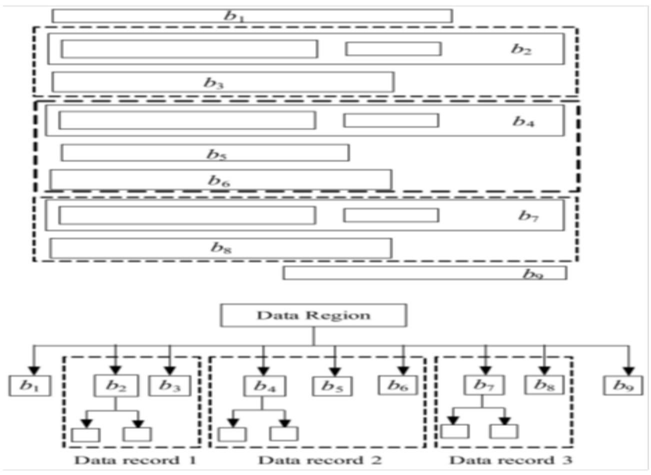

Fig. 3- A general case of data region.

Data Records Extraction

Data record extraction aims to discover the boundary of data records and extract them from the deep Web pages. An ideal record extractor should achieve the following:

- All data records in the data region are extracted and
- For each extracted data record, no data item is missed and no incorrect data item is included.

Data record extraction is to discover the boundary of data records based on the LF and AF features. That is, we attempt to determine which blocks belong to the same data record. We achieve this in the following three phases:

1) Phase 1: Filter out some noise blocks.
2) Phase 2: Cluster the remaining blocks by computing their appearance similarity.
3) Phase 3: Discover data record boundary by regrouping blocks.

Phase 1: Noise Blocks Filtering

Because noise blocks are always at the top or bottom, we check the blocks located at the two positions according to LF1. If a block at these positions is not aligned flush left, it will be removed as a noise block. This step does not guarantee the removal of all noise blocks. For example, in Fig. 5, block b9 can be removed in this step, while block b1 cannot be removed.

Phase 2: Blocks Clustering

The remaining blocks in the data region are clustered based on their appearance similarity. Since there may be three kinds of information in data records, i.e., images, plain text and link text, the appearance similarity between blocks is computed from the three aspects. For images, we care about the size; for plain text and link text, we care about the shared fonts. Intuitively, if two blocks are more similar on image size and font, they should be more similar in appearance.

Phase 3: Blocks Regrouping

The clusters obtained in the previous step do not correspond to data records. On the contrary, the blocks in the same cluster all come from different data records. According to AF2, the blocks in the same cluster have the same type of contents of the data records.

The blocks need to be regrouped such that the blocks belonging to the same data record form a group. Our basic idea of blocks regrouping is as follows: According to CF1, the first data item in each data record is always m andatory. Clearly, the cluster that contains the blocks for the first items has the maximum number of blocks possible; let n be this maximum number. It is easy to see that if a cluster contains n blocks, these blocks contain m andatory data items. Our regrouping method first selects a cluster with n blocks and uses these blocks as seeds to form data records. Next, given a block b, we determine which record b belongs to according to CF2.

For example, suppose we know that title is ahead of author in each record and they belong to different blocks (say an author block and a title block). Each author block should relate to the nearest title block that is ahead of it. In order to determine the order of different semantic blocks, a minimum bounding rectangle is formed for each cluster on the page. By comparing the positions of these rectangles on the page, we can infer the order of the semantics. For example, if the rectangle enclosing all title blocks is higher than the rectangle enclosing the author blocks, then title must be ahead of its corresponding author.

Visual Wrapper Generation

Since all deep Web pages from the same Web database share the same visual template, once the data records and data items on a deep Web page have been extracted, we can use these extracted data records and data items to generate the extraction wrapper for the Web database so that new deep Web pages from the same Web database can be processed using the wrappers quickly without reapplying the entire extraction process. Our wrappers include data record wrapper and data item wrapper. They are the programs that do data record extraction and data item extraction with a set of parameter obtained from sample pages. For each Web database, we use a normal deep Web page containing the maximum number of data records to generate the wrappers. The wrappers of previous works mainly depend on the structures or the locations of the data records and data items in the tag tree, such as tag path. In contrast, we mainly use the visual information to generate our wrappers. Note that some other kinds of information are also utilized to enhance the performances of the wrappers, such as the data types of the data items and the frequent symbols appearing in the data items. But they are easy to obtain from the Web pages. We describe the basic ideas of our wrappers below.

Conclusion and Future Scope

With the flourish of the deep Web, users have a great opportunity to benefit from such abundant information in it. Therefore, it is an important task to extract the structured data from the deep Web pages for later processing. In this paper, we focused on the structured Web data extraction problem, including data record extraction and data item extraction. Based on previous survey, we found that the visual information of Web pages can help us implement Web data extraction. Based on our observations of a large number of deep Web pages, we identified a set of interesting common visual features that are useful for deep Web data extraction. Based on these visual features, we proposed web programming language independent approach to extract structured data from deep Web pages, which can avoid the limitations of previous works. The main trait of this vision-based approach is that it primarily utilizes the visual features of deep Web pages.

Due to the size limitation we are not placing implementation results of proposed work.

References

[1] Liu W., Meng X. and Meng W. (2010) *IEEE Transations on Knowledge and data Engineering*, 22(3), 447- 460.

[2] Arocena G.O. and Mendelzon A.O. (1998) *Int'l Conf. Data Eng. (ICDE)*, 24-33.

[3] Chang C.H., Hsu C.N. and Lui S.C. (2003) *Decision Support Systems*, 35(1), 129-147.

[4] Hammer J., McHugh J. and Garcia-Molina H. (1997) *East-European Workshop Advances in Databases and Information Systems (ADBIS)*, 1-8.

[5] Madhavan J., Jeffery S.R., Cohen S., Dong X.L., Ko D., Yu C. and Halevy A. (2007) *Conf. Innovative Data Systems Research (CIDR)*, 342-350.

[6] Kushmerick N. (2000) *Artificial Intelligence*, 118(1/2), 15-68.
[7] Liu L., Pu C. and Han W. (2000) *Int'l Conf. Data Eng. (ICDE)*, 611-621.