DATA EXTRACTION FROM DYNAMIC WEB PAGES BASED ON VISUAL FEATURES

Prof. Sachin Bojewar, Prof. Varsha Bhosale, Shuveta Chanchlani

Address for Correspondence
Vidyalankar Institute of Technology, Wadala, Mumbai, MS

1. 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 millions according to a recent survey. 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. In this paper, we call this kind of special Web pages deep Web pages. Each data record on the deep Web pages corresponds to an object. In order to ease the consumption by human users, most Web databases display data records and data items regularly on Web browsers.

Finding information about people in the World Wide Web is one of the most common activities of Internet users. Person names, however, are highly ambiguous. In most cases, the results for a person name search are a mix of pages about different people sharing the same name. The user is then forced to either add terms to the query (probably losing recall and focusing on one single aspect of the person), or to browse every document in order to filter the information about the person he/she is actually looking for. In an ideal system the user would simply type a person name, and receive search results clustered according to the different people sharing that name. One particular case of this people-document association task is referred to as personal name resolution. The task is as follows: given a set of documents all of which refer to a particular person name but not necessarily a single individual (usually called referent), identify which documents are associated with each referent by that name. Different methods have been used to represent documents that mention a candidate, including snippets, text around the person name, entire documents, extracted phrases, etc.

2. PROBLEM STATEMENT

Web Search machines are absolutely not new concepts, but typically they’ve included rate limits, strict terms of service regarding the re-ordering and presentation of results, and provided little or no opportunity for monetization. These constraints have limited the innovation and commercial viability of new search solutions. Maximum of the information mining is time consuming as the information originates from multiple sites, which sometimes are not updated. Searching for two people with same adds to complexity while search.

2.1 Existing System and its effect

Searching for information on the Web is not an easy task. Searching for personal information is sometimes even more complicated. Below are several common problems we face when trying to get personal details from the web:

- Majority of the Information is distributed between different sites.
- It is not updated.
- Multi-Referent ambiguity – two or more people with the same name.
- Multi-morphic ambiguity which is because one name may be referred to in different forms.
- In the most popular search engine Google, one can set the target name and based on the extremely limited facilities to narrow down the search, still the user has 100% feasibility of receiving irrelevant information in the output search hits. Not only this, the user has to manually see, open, and then download their respective file which is extremely time consuming. The major reason behind this is that there is no uniform format for personal information.

Maximum of the past work is based on exploiting the link structure of the pages on the web, with hypothesis that web pages belonging to the same person are more likely to be linked together.

2.2 Proposed System

One of the key challenges that needs to be overcome to make the project functionality a reality, is to build an advance query system that is capable of reaching high disambiguation quality. The project work is targeted to design an advance version of the deep extraction tool using Clustering Algorithm. In this research work, the focus is mainly on querying for personal information of scientists and researchers. The user has to set the proper target name for search, which when completed, the user will receive complete PDF and image files based on the key of the search. Each group of information items (cluster) will be defined by its key and the user make the choice. The result page will be produced from the chosen clusters.
For making the search operationally accurate, we will assume the usage of IEEE doc files as they carry a standard format of name, e-mail ID, publication, images, and links to the full images. The visual information of Web pages, which has been introduced above, can be obtained through the programming interface provided by Web browsers. 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.

3. 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.

3.1 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.

3.2 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>
<thead>
<tr>
<th>Font factor</th>
<th>Example</th>
<th>Font factor</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>( (0,0) )</td>
<td>underline</td>
<td>( \Delta )</td>
</tr>
<tr>
<td>face</td>
<td>A(Sans Serif)</td>
<td>italic</td>
<td>( \Delta )</td>
</tr>
<tr>
<td>color</td>
<td>( (red) )</td>
<td>weight</td>
<td>( \Delta )</td>
</tr>
<tr>
<td>strikethrough</td>
<td>( \Delta )</td>
<td>frame</td>
<td>( \Delta )</td>
</tr>
</tbody>
</table>

4. THE VIPS ALGORITHM
In the VIPs algorithm, the vision-based content structure of a page is deduced by combining the DOM structure and the visual cues. First, DOM structure and visual information, such as position, back ground color, font size, font weight, etc., are obtained from a web browser. Then, from the root node, the visual block extraction process is started to extract visual blocks of the current level from the DOM tree based on visual cues. Every DOM node is checked to judge whether it forms a single block or not. If not, its children will be processed in the same way. When all blocks of the current level are extracted, they are put into a pool. Visual separators among these blocks are identified and the weight of a separator is set based on properties of its neighboring blocks. After constructing the layout hierarchy of the current level, each newly produced visual blocks is checked to see whether or not it meets the granularity requirement. If no, this block will be further partitioned. After all blocks are processed, the final vision-based content structure for the web page is outputted. Below we introduce the visual block extraction, separator detection and content structure construction phases respectively.

4.1 Visual Block Extraction
In this phase, we aim at finding all appropriate visual blocks contained in the current sub-tree. In general, every node in the DOM tree can represent a visual block. However, some “huge” nodes such as \(<TABLE>\) and \(<P>\) are used only for organization purpose and are not appropriate to represent a single visual block. In these cases, the current node should be further divided and replaced by its children. On the other hand, we may not extract all leaf nodes in the DOM tree due to their high volume.

At the end of this step, for each node that represents a visual block, its DoC value is set according to its intra visual difference. This process is iterated until all appropriate nodes are found to represent the visual blocks in the web page.

Some important cues are used to produce heuristic rules in the algorithm are:

- Tag cue: Tags such as \(<HR>\) are often used to separate different topics from visual perspective. Therefore we prefer to divide a DOM node if it contains these tags.
- Color cue: We divide a DOM node if its background color is different from one of its children’s.
- Text cue: If most of the children of a DOM node are Text nodes (i.e., no tags surround them), we do not divide it.
- Size cue: We prefer to divide a DOM node if the standard deviation of size of its children is larger than a threshold.

4.2 Visual Separator Detection
After all blocks are extracted, they are put into a pool for visual separator detection. Separators are horizontal or vertical lines in a web page that visually cross with no blocks in the pool. From a visual perspective, separators are good indicators for discriminating different semantics within the page. A visual separator is represented by a 2-tuple: \((Ps, Pe)\), where \(Ps\) is the start pixel and \(Pe\) is the end pixel. The width of the separator is calculated by the difference between these two values.

5. PROJECT IMPLEMENTATION
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: 1) all data records in the data region are extracted and 2) for each extracted data record, no data item is missed and no incorrect data item is included. Instead of extracting data records from the deep Web page directly, we first locate the data region, and then, extract data records from the data region. PF1 and PF2 indicate that the data records are the primary content on the deep Web pages and the data region is centrally located on these pages. The data region corresponds to a block in the Visual Block tree. We locate the data region by finding the block...
that satisfies the two position features. Each feature can be considered as a rule or a requirement. The first rule can be applied directly, while the second rule can be represented by \( \text{area}_b > \text{area}_{\text{page}} - T_{\text{region}} \), where \( \text{area}_b \) is the area of block \( b \), \( \text{area}_{\text{page}} \) is the area of the whole deep Web page, and \( T_{\text{region}} \) is a threshold. The threshold is trained from sample deep Web pages. If more than one block satisfies both rules, we select the block with the smallest area. Though very simple, this method can find the data region in the Visual Block tree accurately and efficiently. We have implemented an operational deep Web data extraction system for ViDE based on the techniques we proposed. Our experiments are done on a Pentium 4 1.9 GH, 512 MB PC.

6. CONCLUSION

The World Wide Web is a rapidly growing and changing information source. Due to the dynamic nature of the Web, it becomes harder to find relevant and recent information. We present a new model and architecture of the Web Crawler using multiple HTTP connections to WWW. The multiple HTTP connection is implemented using multiple threads and asynchronous downloader module so that the overall downloading process is optimized. The user specifies the start URL from the GUI provided. It starts with a URL to visit. As the crawler visits the URL, it identifies all the hyperlinks in the web page and adds them to the list of URLs to visit, called the crawl frontier. URLs from the frontier are recursively visited and it stops when it reaches more than five level from every home pages of the websites visited and it is concluded that it is not necessary to go deeper than five levels from the home page to capture most of the pages actually visited by the people while trying to retrieve information from the internet. The web crawler system is designed to be deployed on a client computer, rather than on mainframe servers which require a complex management of resources, still providing the same information data to a search engine as other crawlers do.

7. FUTURE DEVELOPMENT

Web Crawler forms the back-bone of applications that facilitate Web Information Retrieval. In this paper we have presented the architecture and implementation details of our crawling system which can be deployed on the client machine to browse the web concurrently and autonomously. It combines the simplicity of asynchronous downloader and the advantage of using multiple threads. It reduces the consumption of resources as it is not implemented on the mainframe servers as other crawlers also reducing server management. The proposed architecture uses the available resources efficiently to make up the task done by high cost mainframe servers. A major open issue for future work is a detailed study of how the system could become even more distributed, retaining though quality of the content of the crawled pages. Due to dynamic nature of the Web, the average freshness or quality of the page downloaded need to be checked, the crawler can be enhanced to check this and also detect links written in JAVA scripts or VB scripts and also provision to support file formats like
XML, RTF, PDF, Microsoft word and Microsoft PPT can be done.

REFERENCES