AUTOMATIC TEMPLATE EXTRACTION FROM HETEROGENEOUS WEB PAGES

First Pratap Bangosavi, RKDF Institute of Science & Technology, Bhopal-462047; Second Jasvindar Pal Singh, Assistance Professor, RKDF Institute of Science & Technology, Bhopal-462047

Abstract

The templates provide readers easy access to the contents guided by consistent structures. However, for machines, the templates are considered harmful since they degrade the accuracy and performance of web applications due to irrelevant terms in templates. The World Wide Web is a vast and most useful collection of information. To achieve high productivity in publishing the web pages are automatically evaluated using common templates with contents. The pages belong to each group should possess the same structure. This saves the time to find out best templates from a large number of web document and also saves the memory which is required to find out the best template structure. We develop a novel goodness measure with its fast approximation for clustering and provide comprehensive analysis of our algorithm. Our experimental results with real-life data sets confirm the effectiveness and robustness of our algorithm compared to the state of the art for template detection algorithms.

Introduction

WORLD Wide Web (WWW) is widely used to publish and access information on the Internet. In order to achieve high productivity of publishing, the webpages in many websites are automatically populated by using common templates with contents. For human beings, the templates provide readers easy access to the contents guided by consistent structures even though the templates are not explicitly announced. However, for machines, the unknown templates are considered harmful because they degrade the accuracy and performance due to the irrelevant terms in templates. Thus, template detection and extraction techniques have received a lot of attention recently to improve the performance of web applications, such as data integration, search engines, classification of web documents and so on [3], [4], [6], [7], [8], [10]. For example, biogenic data are published on the Internet by many organizations with different formats and scientists want to integrate these data into a unified database.

For price comparison services, the price information is gathered from various Internet marketplaces. Good template extraction technologies can significantly improve the performance of these applications. To overcome the limitation of the techniques with the assumption that the web documents are from a single template, the problem of extracting the templates from a collection of heterogeneous web documents, which are generated from multiple templates, was also studied. In this problem, clustering of web documents such that the documents in the same group belong to the same template is required, and thus, the correctness of extracted templates depends on the quality of clustering. Since an HTML document can be naturally represented with a Document Object Model (DOM) tree, web documents are considered as trees and many existing similarity measures for trees have been investigated for clustering [6], [9], [11]. However, clustering is very expensive with tree-related distance measures. For instance, tree-edit distance has at least O time complexity [6], [9], where n1 and n2 are the sizes of two DOM trees and the sizes of the trees are usually more than a thousand. Thus, clustering on sampled web documents is used to practically handle a large number of web documents. In Fig. 1, given two pages look clearly different. However, their URLs are identical except the value of a layout parameter. If we use only URLs to group pages, these pages from the different templates will be included in the same cluster.

Fig. 1. Different templates of the same URL.

However, it is not easy to select proper training data of small size, since we do not have any knowledge about given data in advance. Moreover, it is hard to decide how many clusters are to be generated from the web documents. In [6], they empirically suggested 80 percent similarity threshold within a cluster but it does not work for all the time. In [11], it is assumed that labeled training data are given for clustering. However, this assumption is not valid either in many cases. In this paper, in order to alle-
viatiate the limitations of the state-of-the-art technologies, we investigate the problem of detecting the templates from heterogeneous web documents and present novel algorithms called TEXT (automatic template extraction). We propose to represent a web document and a template as a set of paths in a DOM tree. As validated by the most popular XML query language XPATH [2], paths are sufficient to express tree structures and useful to be queried. By considering only paths, the overhead to measure the similarity between documents becomes small without significant loss of information. For example, let us consider simple HTML documents and paths in Fig. 2 and Table 1. We will formally define the paths later. Document d1 is represented as a set of paths \{p1; p2; p3; p4; p6\} and the template of both d1 and d2 is another set of paths \{p1; p2; p3; p4\}.

![Fig. 2. Simple web documents. (a) Document d1. (b) Document d2. (c) Document d3. (d) Document d4.](image)

TABLE 1
Paths of Tokens and Their Supports

<table>
<thead>
<tr>
<th>ID</th>
<th>Path</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>Document{html}</td>
<td>4</td>
</tr>
<tr>
<td>p2</td>
<td>Document{html}{body}</td>
<td>4</td>
</tr>
<tr>
<td>p3</td>
<td>Document{html}{body}{h1}</td>
<td>3</td>
</tr>
<tr>
<td>p4</td>
<td>Document{html}{body}{br}</td>
<td>3</td>
</tr>
<tr>
<td>p5</td>
<td>Document{html}{body}{List}</td>
<td>3</td>
</tr>
<tr>
<td>p6</td>
<td>Document{html}{body}{h1}{Tech}</td>
<td>1</td>
</tr>
<tr>
<td>p7</td>
<td>Document{html}{body}{h1}{World}</td>
<td>1</td>
</tr>
<tr>
<td>p8</td>
<td>Document{html}{body}{h1}{Local}</td>
<td>1</td>
</tr>
</tbody>
</table>

Related Work

The template extraction problem can be categorized into two broad areas. The first area is the site-level template detection where the template is decided based on several pages from the same site. We also adopt this observation and consider every word equally in our solution. However, they detect elements of a template by the frequencies of words but we consider the MDL principle as well as the frequencies to decide templates from heterogeneous documents. While HTML documents are semi-structured, XML documents are well structured, and all the tags are always a part of a template. The solutions for XML documents fully utilize these properties. In the problem of the template extraction from heterogeneous document, how to partition given documents into homogeneous subsets is important. Reis et al. [6] used a restricted tree-edit distance to cluster documents and, in [11], it is assumed that labeled training data are given for clustering.

**A. HTML Documents and Document Object Model**

The DOM defines a standard for accessing documents, like HTML and XML [1]. The DOM presents an HTML document as a tree structure. The entire document is a document node, every HTML element is an element node, the texts in the HTML elements are text nodes, every HTML attribute is an attribute node, and comments are comment nodes. However, we do not distinguish the type of nodes, since, as defined in [3], any type of node can be a part of a template in our problem. For instance, the DOM tree of a simple HTML document d2 in Fig. 2b is given in Fig. 3. For a node in a DOM tree, we denote the path of the node by listing nodes from the root to the node in which we use “n” as a delimiter between nodes. For example, in the DOM tree of d2 in Fig. 3, the path of a node “World” is “Documenting html in body in h1 in World.”

![Fig. 3. DOM Tree of d2 in Fig. 2.](image)

**B. Essential Paths and Templates**

Given a web document collection D={d1; d2; . . .; dN}, we define a path set PD as the set of all paths in D. Note that, since the document node is a virtual node shared by every document, we do not consider the path of the document node in PD. The support of a path is defined as the number of documents in D, which contain the path. For each document di, we provide a minimum support threshold tdi. Notice that the thresholds tdi and t dj of two distinct documents di and dj, respectively, may e different. If a
path is contained by a document $d_i$ and the support of the path is at least the given minimum support threshold $t_d$, the path is called an essential path of $d_i$. We denote the set of essential paths of an HTML document $d_i$ by $E(d_i)$. For a web document set $D$ with its path set $PD$, we use a $|PD| \times |D|$ matrix $M_E$ with 0/1 values to represent the documents with their essential paths. The value at a cell $(i, j)$ in the matrix $M_E$ is 1 if a path $p_i$ is an essential path of a document $d_j$. Otherwise, it is 0.

$$
M_E = \begin{bmatrix}
  d_1 & d_2 & d_3 & d_4 \\
  p_1 & 1 & 1 & 1 & 1 \\
  p_2 & 1 & 1 & 1 & 1 \\
  p_3 & 1 & 1 & 1 & 1 \\
  p_4 & 1 & 1 & 1 & 0 \\
  p_5 & 0 & 1 & 1 & 0 \\
  p_6 & 0 & 0 & 0 & 0 \\
  p_7 & 0 & 0 & 0 & 0 \\
  p_8 & 0 & 0 & 0 & 0 
\end{bmatrix}
$$

### System Architecture

In this project we are providing different web pages as input to our system. Each web page has different or may be same document structure. After that we are parsing these web documents into an xml document using DOM model. After that we are finding out the paths or flow or document structure using its tag entry in the XML document. After that we are applying the MinHash algorithm to find out best pair pages from given input Pages. Then we are classifying these best pair pages into different groups. We are recommending these groups to the user. This saves the time to find out best templates from large no of web document and also save the memory i.e. required to find out the best template structure.

### Algorithm Required

Algorithm: Min-Hash

**Input:** Web Pages

1. GetBestPair (Clusters, Documents )
2. Initial $C$ = {cluster1, cluster2….documentN} // here initially no of clusters= no of documents
3. For each pair clusterI,clusterJ of Clusters in C
4. 1.3) min MDLCost=0
5. 1.4) MDLCost=calculate MDLCost (clusterI, clusterJ)
6. If (min MDLCost> MDLCost)
7. Min MDLCost==MDLCost;
8. Store pair (clusterI, clusterJ);
9. 1.5) cluster pages which having less MDLCost than other pair
10. 1.6) update Cluster Set C by merging best pair in one cluster.

**Function MDLCost** (clusterI, clusterJ)

1. Get all paths from each cluster template;
2. Calculate $M_t$, $M_d$, $H(x)$ using given formula.
3. Calculate $L(c) = L(M_t) + L(M_d) + L(M_{delta})$;
4. Return MDLCost

### Acknowledgments

We are so thankful to IJATER Journal for the support to develop this document. And My college also help to develop this paper publication.

### References


Biographies

FIRST PRATAP BANGOSAVI received the B.E. degree in Computer Science & Engineering from the University of Shivaji University, Kolhapur, Maharashtra, in 2008, the M.Tech degree in Software Engineering from the University of RGPV, Bhopal, Madhya Pradesh, Currently, I am Student of RKDF M.Tech Second year.
Pratap.bangosavi84@gmail.com.