Volume : 2 | Issue : 9 | September 2013 • ISSN No 2277 - 8179

Research Paper

Template Detection Technique From Assorted Web Pages

Engineering

KEYWORDS : HTML, Document, DOM Level, Semantics

Marriboyina Rajendra — MTech(Software Engineering) Student , SRI MITTAPALLI COLLEGE OF ENGINEERING,TUMMALAPALEM

S. Suresh Babu — Asst.Professor , Dept of CSE, SRI MITTAPALLI COLLEGE OF ENGINEERING,TUMMALAPALEM

ABSTRACT

Now a Days unstructured and/or semi-structured machine-readable document automatically plays a major role in Extracting structured information. To achieve publishing productivity many websites are using common templates with contents to populate the information and the major resource as we all know is WWW. Performance of search engine, clustering and classification of web documents got lot of Concentration for Template detection technique, as templates degrade the performance and accuracy of web application for machines because of irrelevant template terms. In this paper, we present novel algorithms for extracting templates from a large number of web documents which are generated from heterogeneous templates. Using the similarity of underlying template structures in the document we cluster the web documents so that template for each cluster is extracted simultaneously. Among the Template detection algorithms we can justify our algorithm is most efficient one.

I .INTRODUCTION

The HTML DOM follows a naming convention for properties, methods, events, collections, and data types. All names are defined as one or more English words concatenated together to form a single string. Properties and Methods .The property or method name starts with the initial keyword in lowercase, and each subsequent word starts with a capital letter. For example, a property that returns document meta information such as the date the file was created might be named "fileDateCreated". In the ECMAScript binding, properties are exposed as properties of a given object. In Java, properties are exposed with get and set methods. Non-HTML 4.0 interfaces and attributes. While most of the interfaces defined below can be mapped directly to elements defined in the HTML 4.0 Recommendation, some of them cannot. Similarly, not all attributes listed below have counterparts in the HTML 4.0 specification (and some do, but have been renamed to avoid conflicts with scripting languages). Interfaces and attribute definitions that have links to the HTML 4.0 specification have corresponding element and attribute definitions there; all others are added by this specification, either for convenience or backwards compatibility with "DOM Level 0" implementations.

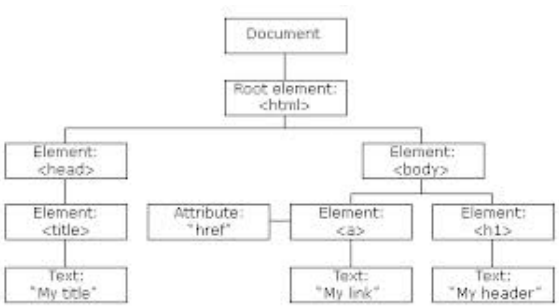


Fig.1 HTML Document Representation

HTML elements form the building blocks of all websites. HTML allows images and objects to be embedded and can be used to create interactive forms. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. It can embed scripts written in languages such as JavaScript which affect the behavior of HTML web pages.Web browsers can also refer to Cascading Style Sheets (CSS) to define the appearance and layout of text and other material. The W3C, maintainer of both the HTML and the CSS standards, encourages the use of CSS over explicit presentational HTML markup.

In this paper 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 , 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.

II. BACKGROUND 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. Crescenzi et al. [1] studied initially the data extraction problem and Yossef and Rajagopalan [4] introduced the template detection problem. Previously, only tags were considered to find templates but Arasu and Garcia-Molina [3] observed that any word can be a part of the template or contents. 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 heterogenous documents. Vieira et al. [2] suggested an algorithm considering documents as trees but the operations on trees are usually too expensive to be applied to a large number of documents. Zhao et al. [2], [2] concentrated on the problem of extracting result recordsfrom search engines. For XML documents, Garofalakis et al. [4] solved the problem of DTD extraction from multiple XML documents. While HTML documents are semistructured, 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. [8] used a restricted tree-edit distance to cluster documents and, in [6], it is assumed that labeled training data are given for clustering. However, the treeedit distance is expensive and it is not easy to select good training pages. Crescenzi et al. [1] focused on document clustering without template extraction. They targeted a site consisting of multiple templates. From a seed page, web pages are crawled by following internal links and the pages are compared by only their link information. However, if web pages are collected without considering their method, pages from various sites are mixed in the collection and their algorithm should repeatedly be executed for each site. Since pages crawled from a site can be different by the objectivity of each crawler, their algorithm may require additional crawling on the fly. The other area is the page-level template detection where the template is computed within a single document. Lerman et al. [6] proposed systems to identify data records in a document and extract data items from them. Zhai and Liu [3] proposed an algorithm to extract a template using not only structural information, but also visual layout information. Chakrabarti et al. [6] solved this problem by using an isotonic smoothing score assigned by a classifier. Since the problem formulation of this area is far from ours, we do not discuss it in detail. Our algorithms

Volume : 2 | Issue : 9 | September 2013 • ISSN No 2277 - 8179

to be presented later represent web documents as a matrix and find clusters with the matrix.

III. THE PROPOSED FRAMEWORK

Our template detection method is based on the repetition of text segments which are text nodes in DOM trees of web pages. We the contents and DFs of text segments. We should note that when talking about the DFs of text segments, we must first clarify when two text segments are considered to be the same. In our framework, two text segments are same only when their contents are literally equal and they have the same DOM path. Whenever a new page is available, it will be passed through four steps:

1) Page segmentation,
2) Text segment table expansion,
3) Template detection, and
4) Text segment table shrinkage.

1. Page Segmentation

The segmentation process contains two steps: a) A web page is divided into multiple blocks. Currently, we choose some html tags that usually determine the page layout as separators, these html tags are <TABLE>, <DIV>, etc.

b) Then each block is further divided into text segments by html tags, process instructions, and html comments.

2. Text Segment Table Expansion

After page segmentation, text segments are used to update the text segment table. If a text segment already exists in the table, the DF of it will be increased by one. Otherwise, the text segment will be inserted into the table and its DF is initialized to one.

3. Template Detection

Template detection occurs in block level. We search every text segment of a block in the text segment table, checking whether it is a template segment. We define template segments as text segments whose DFs are larger than or equal to 5. We can then calculate the template ratio of a block: template ratio = _lengths of template segments _lengths of all text segments If the template ratio of a block is larger than 0.7, we label the block as a template block.

4. Text Segment Table Shrinkage

The text segment table will consume more and more storage if only the expansion step is applied. To control the storage use, we need to delete some text segments. The cost of deleting a text segment is defined as the times to classify a template segment as non-template segment be- cause of the deletion. For example, if a template segment appears after its deletion, it will be recognized as non-template segment. The cost of deleting a text segment is related to the DF and the future occurring times of the text segment. To minimize the cost of deletion, we allow text segments that have larger DFs to live longer than those with smaller DFs, because the former are more likely to occur in the future. We should note that we don't use the publish times or crawling times as the timestamps of pages and blocks, but we assign every page a page number and use it as the timestamp.

IV. PROPOSED ALGORITHM

The TEXT-MDL algorithm

The theory of minimum message length (MML) and minimum description length (MDL) first appears in the computation complexity community [4] then in the categorization community [5]. Its application in data mining community is the work of climate data segmentation [6], trajectory clustering [7], and social network mining [8]. So far, to the best of our knowledge, our work segmenting time-series stream with MDLMML is the work with the most features.

```
algorithm TEXT-MDL(D /* a set of document */)
begin
1.  C := {c_1, c_2, ..., c_n} with c_i = (E(d_i), {d_i});
2.  (c_i, c_j, c_k) := GetBestPair(C);
3.  /* Let c_i and c_j be the best pair for merging */
4.  /* Let c_k be a new cluster made by merging c_i and c_j */
5.  while (c_i, c_j, c_k) is not empty do {
6.      C := C - {c_i, c_j} ∪ {c_k};
7.      (c_i, c_j, c_k) := GetBestPair(C);
8.  }
9.  return C
end

procedure GetBestPair(C /* a clustering model */)
begin
1.  MDLcost_min := ∞;
2.  for each pair (c_i, c_j) of clusters in C do {
3.      (MDLcost, c_k) := GetMDLCost(c_i, c_j, C);
4.      /* GetMDLCost returns the optimal MDL cost
5.      when c_k is made by merging c_i and c_j */
6.      if MDLcost < MDLcost_min then {
7.          MDLcost_min := MDLcost;
8.          (c_i^B, c_j^B, c_k^B) := (c_i, c_j, c_k);
9.      }
10. }
11. return (c_i^B, c_j^B, c_k^B);
end
```

GetMDLCost/GetBestPair Algorithm

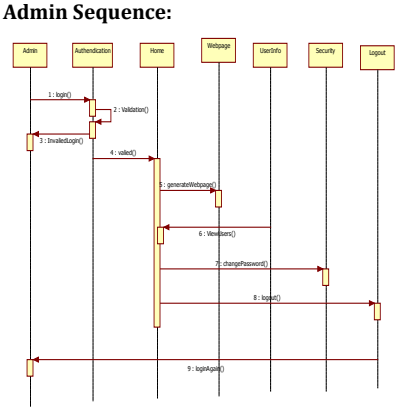
```
Procedure GetInitBestPair(C)
begin
1.  Merge all clusters with the same signature of MinHash;
2.  MDL_min := ∞;
3.  for each c_i in C do {
4.      N := clusters with the maximal Jaccard's coeff. with c_i;
        /* If the maximal Jaccard's coefficient is 0, N is ∅ */
5.      for each c_j in N do {
6.          (MDL_tmp , c_k) := GetHashMDLCost(c_i, c_j, C);
7.          if MDL_tmp < MDL_min then {
8.              MDL_min := MDL_tmp;
9.              (c_i^B, c_j^B, c_k^B) := (c_i, c_j, c_k);
10.         }
11.     }
12. }
13. return (c_i^B, c_j^B, c_k^B);
end

Procedure GetHashBestPair(c_k, C)
begin
1.  (c_i^B, c_j^B) := the current best pair;
2.  c_k^B := a cluster made by merging c_i^B and c_j^B;
3.  MDL_min := the current best approximate MDL cost;
4.  N := clusters with the maximal Jaccard's coeff. with c_k;
    /* If the maximal Jaccard's coefficient is 0, N is ∅ */
5.  for each c_t in N do {
6.      (MDL_tmp , c_tmp) := GetHashMDLCost(c_k, c_t, C);
7.      if MDL_tmp < MDL_min then {
8.          MDL_min := MDL_tmp;
9.          (c_i^B, c_j^B, c_k^B) := (c_k, c_t, c_tmp);
10.     }
11. }
12. return (c_i^B, c_j^B, c_k^B);
end
```

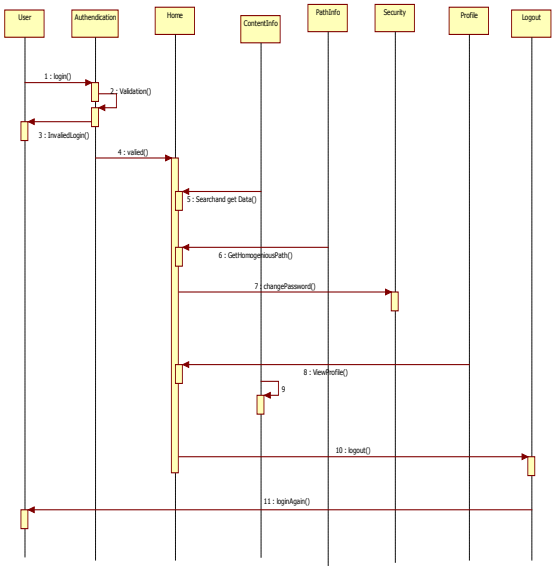
V. SYSTEM ANALYSIS & DESIGN

Sequence diagrams belong to a group of UML diagrams called Interaction Diagrams. Sequence diagrams describe how objects interact over the course of time through an exchange of messages.

Admin Sequence:



User Sequence:



VI. RESULTS

CONCLUSION

From heterogeneous web documents we have introduced an approach of template detection. Using MDL Principle we have managed number of clusters and selected Good partitioning from all Possible Partitions of documents. To speedup the Clustering process we have introduced Extended MinHash Technique .Using Real Life data sets we have experimentally proved that the efficiency of our proposed algorithm

REFERENCE

[1] J. Rissanen, Stochastic Complexity in Statistical Inquiry. World Scientific, 1989. | [2] K. Vieira, A.S. da Silva, N. Pinto, E.S. de Moura, J.M.B. Cavalcanti, and J. Freire, "A Fast and Robust Method for Web Page Template detection and Removal," Proc. 15th ACM Int'l Conf. Information and Knowledge Management (CIKM), 2006. | 3. B. Long, Z. Zhang, and P.S. Yu, "Co-Clustering by Block Value Decomposition," Proc. ACM SIGKDD, 2005. | 4. K. Vieira, A.S. da Silva, N. Pinto, E.S. de Moura, J.M.B. Cavalcanti,and J. Freire, "A Fast and Robust Method for Web Page Template Detection and Removal," Proc. 15th ACM Int'l Conf. Information and Knowledge Management (CIKM), 2006. | 5. H. Zhao, W. Meng, and C. Yu, "Automatic Extraction of Dynamic Record Sections from Search Engine Result Pages," Proc. 32nd Int'l Conf. Very Large Data Bases (VLDB), 2006. | 6. Document Object Model (dom) Level 1 Specification Version 1.0, http://www.w3.org/TR/REC-DOM-Level-1, 2010. | 9. D. Chakrabarti, R. Kumar, and K. Punera, "Page-Level Template Detection via Isotonic Smoothing," Proc. 16th Int'l Conf. World Wide Web (WWW), 2007. | 10. T.M. Cover and J.A. Thomas, Elements of Information Theory. Wiley Interscience, 1991. |