AUTOMATIC TEMPLATE DETECTION USING NOVEL APPROACH

T.PRATHIBHA, B.SUKUMAR, A.SRIRAM

M.Tech Student, CSE Department, SSJ Engineering College, AP, India,

Assistant Professor, CSE Department, SSJ Engineering College, AP, India,

Assistant Professor, IT Department, ANURAG Group of Institution, AP, India.

Abstract

The templates provide readers easy access to the contents guided by consistent structures. Thus, template detection techniques have received a lot of attention recently to improve the performance of search engines, clustering, and classification of web documents. In this paper, we present novel algorithms for detecting and extracting templates from a large number of web documents which are generated from heterogeneous templates. We cluster the web documents based on the similarity of underlying template structures in the documents so that the template for each cluster is extracted simultaneously. 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.

Keywords: *Automatic Template Detection, Content Extraction, Clustering, M.D.L Cost*.

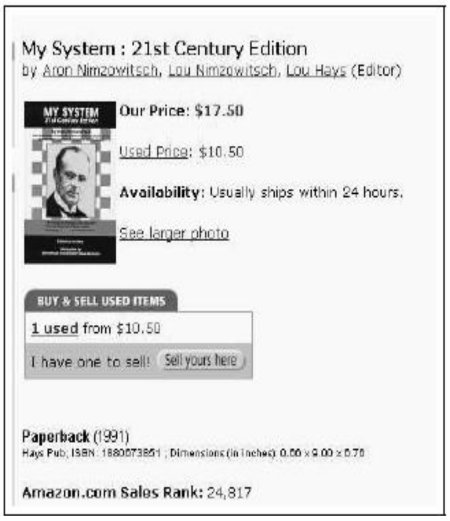
1. Introduction

World Wide Web is the most useful source of information. In order to achieve high productivity of publishing, the web pages in many websites are automatically populated by using the common templates with contents. 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. Thus, template detection techniques have received a lot of attention recently to improve the performance of search engines, clustering, and classification of web documents template material is common content or formatting that appears on multiple pages of a site. Almost all pages on the web today contain template material to a greater or lesser extent. Common examples include navigation sidebars containing links along the left or right side of the page; corporate logos that appear in a uniform location on all pages; standard background colors or styles; headers or dropdown menus along the top with links to products, locations, and contact information; banner advertisements; and footers containing links to homepages or copyright information. The template mechanism is used to support many purposes, particularly navigation, presentation, and branding.

2. Project Policy

Search engines are often compared based on how much of the Web they index. For example, Google claims that they currently index 8 billion Web pages, while Yahoo states its index covers 20 billion pages. Unfortunately, comparing search engines based on the sheer number of indexed pages is often misleading because of the unbounded number of pages available on the Web. For example, consider a calendar page that is generated by a dynamic Web site. Since such a page often has a link to the "next-day" or "next-month" page, a Web crawler1 can potentially download an unbounded number of pages from this single site by following an unbounded sequence of next-day links. Thus, 10 billion pages indexed by one search engine may not have as many "interesting"pages as 1 billion pages of another search engine if most of its pages came from a single dynamic Web site.
The World Wide Web is a vast repository of information. The amount of data stored in electronic

databases accessible to users through search forms and dynamically generated Web pages, the so-called hidden Web, dwarfs the amount of information available on static Web pages. Unfortunately, most of this information is presented in a form accessible only to a human user, e.g., list or tables that visually lay out relational data. Although newer technologies, such as XML and the Semantic Web, address this problem directly, only a small fraction of the information on the Web is semantically labeled. The overwhelming majority of the available data has to be accessed in other ways. Web wrappers are popular tools for efficiently extracting information from Web pages. Much of the research in this area over the last decade has been concerned with quick and robust construction of Web wrappers, usually with the help of machine learning techniques. Because even the most advanced of such systems learn correct wrappers from examples provided by the user, the focus recently has been on minimizing the number of examples the user has to label, e.g., through active learning. Still, even when user effort is significantly reduced, the amount and the rate of growth of information on the Web will quickly overwhelm user resources. Maintaining wrappers so that they continue to extract information correctly as Web sites change requires significant effort, although some progress has been made on automating this task. Heuristic techniques that may work in one information domain are unlikely to work in another. A domain-independent, fully automatic solution that requires no user intervention is the Holy Grail of information extraction from the Web. Despite the inherent difficulty of the problem, there are general principles and algorithms that can be used to automatically extract data from structured web sites. In this project, we present new novel techniques that are applicable to a broad range of hidden Web sources.



My System : 21st Century Edition
by Aron Nimzowitsch, Lou Nimzowitsch, Lou Hays (Editor)

Our Price: $17.50

Used Price: $10.50

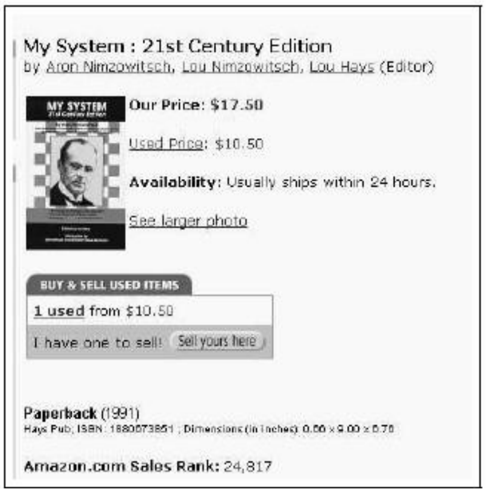
Availability: Usually ships within 24 hours.

See larger photo

BUY & SELL USED ITEMS
1 used from $10.50
I have one to sell! Sell yours here

Paperback (1991)
Hays Pub; ISBN: 1880673851 ; Dimensions (in inches): 0.66 x 9.00 x 6.78

Amazon.com Sales Rank: 24,817



My System : 21st Century Edition
by Aron Nimzowitsch, Lou Nimzowitsch, Lou Hays (Editor)

Our Price: $17.50

Used Price: $10.50

Availability: Usually ships within 24 hours.

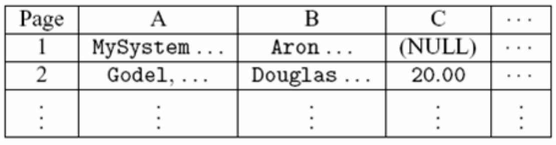
See larger photo

BUY & SELL USED ITEMS
1 used from $10.50
I have one to sell! Sell yours here

Paperback (1991)
Hays Pub; ISBN: 1880673851 ; Dimensions (in inches): 0.66 x 9.00 x 6.78

Amazon.com Sales Rank: 24,817

Fig.1 Sample Amazon Page



Page	A	B	C	···
1	MySystem ...	Aron ...	(NULL)	···
2	Godel, ...	Douglas ...	20.00	···
⋮	⋮	⋮	⋮	⋮

Fig.2 Extracted data

3. Existing System

Table Extraction from HTML Documents

Existing approaches to extracting table data from Web documents can be classified as heuristic or machine learning. Heuristic approaches to detecting tables and record boundaries in Web documents include using the Document Object Model (DOM) and other features to identify tables. Domain-specific heuristic rules that rely on features such as percent signs and date/time formats have also been tried successfully.

Table Extraction from Plain Text

Automatic table extraction from plain text documents is a line of research parallel to the work on HTML table extraction. There are differences between plain text and HTML tables that make the two fundamentally different problems. Plain text documents use white space and new line for the purpose of formatting tables: new lines are used to separate records and white spaces are used to separate columns, among other purposes. Record segmentation from plain text documents is, therefore, a much easier task. Closely linking format and content in plain text documents also gives rise to new challenges. In plain text tables, a long attribute value that may not fit in a table cell will be broken between two lines, creating a non-locality in a text stream. An automatic algorithm will have to associate part of a string with another string that will appear arbitrarily later in the text stream. This problem does not usually arise in HTML documents. On the other hand, HTML tables vary widely in their layout and formatting conventions, making it difficult to rely on any set of features to be good row or column separators. Although one may employ the same set of tools for extracting from HTML and plain text tables, specialized algorithms that address the conventions of each domain may outperform a general algorithm.

DOM based algorithm

This algorithm uses the DOM structure of the pages on a website by searching for nodes of the DOM tree that are repeated across multiple pages on the website. It is based on the work of Rajagopalan and Bar-Yossef and Yi and Liu, but contains simplifications from those techniques. Construction of the DOM tree for a page requires that the page first be cleaned. This is a substantial problem on the Web due to the diverse set of languages, authors, and tools; and also due to the excellent efforts of web browsers to render badly-formed HTML correctly. We modified an existing HTML parsing and cleaning library called HyParSuite to address this problem, maintaining offsets to nodes in the original unclean page so that the links and text inside and outside templates may be extracted later. The algorithm then operates in two passes.

First pass: The first pass iterates over all the pages in the website and dumps information about all the DOM nodes in a page. This information consists of the hash of the content of the node (template-hash) and the start and end offsets into the original file. The template-hash is calculated using the HTML content within the node's start and end tags and DOM node's name, attributes, and their values. For example, consider the following HTML substructure: <td><a href='...'>Click here</a> to visit ...</td> This structure consists of four HTML nodes. The topmost node is the <td> node. The template-hash of this node will be computed from the entire HTML string. The <a> tag is a child of the <td> node and its template-hash will be calculated using the the contents between the <a> and </a> tags inclusive of the tags. Text nodes are constructed for stretches of text in HTML files and the above example consists of two text nodes.

Thus the template-hash is a compressed representation of the HTML tag and its contents. Counting the number of times a template-hash is encountered in a website tells us the number of times a specific HTML node is seen. Hence, the first pass keeps track of the number of times each template hash has been seen in the website and passes this information to the second pass.

Secondpass: The second pass then scans this information and computes a set of template-nodes for each page. A HTML node in a particular page is said to be a template node if the following conditions are met: first, the occurrence count of the node's template-hash is within a specified threshold; and second, the node is not a child of any other template-node.

Sibling template nodes are then coalesced to produce the templates on a page. The coalescing process permits small gaps of changing content in the final templates produced. This is useful for templates with dynamic content, where small portions of the template content changes while the essential HTML and text structure remains the same. Parameter settings: The DOM-based algorithm is parameterized by the upper and lower thresholds on the number of occurrences of template-nodes. A lower-threshold value of 1 will cause the entire web page to be regarded as a single template, as the root of the page always occurs at least once. The upper-threshold parameter prevents the algorithm from detecting extremely small HTML constructs like <BR> as templates just because they are fairly common in HTML files. Other than removing small commonly occurring HTML nodes from consideration, the upper-threshold does not have significant impact on the quality of templates detected.

Text based algorithm

The text-based algorithm does not make use of HTML structural information. The page is pre-processed to remove all HTML tags, comments, and text within <script> tags. The resulting detagged content is typically 2-3 times smaller than the original HTML. The algorithm operates henceforth on this representation. The algorithm detects templates using a two-pass sliding window controlled by four parameters: a window size W, a fragment frequency threshold F, a sampling density D, and a page sample size P. All are described below in more detail.

First pass: In the first pass, P pages are sampled uniformly at random from the crawled pages of the site1 and a window of size W is slid over the text of those pages. At each offset, a counter is incremented for the fragment contained in the window. Those fragments which occur at least F times in the sample are passed to the second pass. For efficiency, we introduce the sampling density parameter D in the first pass. A counter for a fragment is only kept if the hash of the fragment is zero modulo D. Thus, only 1 in every D fragments will be considered, but the down sampling is performed such that if a certain fragment is counted on one page, it will be counted on all pages. Other down sampling mechanisms, such as retaining every Dth fragment, do not have this essential property. We choose D in order to increase the likelihood that after the filtering process concludes, consecutive fragments are contiguous. A coalescing process in the second pass ensures that the total volume of template text is counted correctly. A value of D = 0 in the experiments means all fragments are used. Second pass: In the second pass, each page is scanned for these frequent fragments, and overlapping or contiguous fragments are coalesced into a single template. At the end of the second pass, we have a set of template hashes which are either individual or coalesced fragments. These hashes are stored in a hash table, so that a new page can be broken into fragments and scanned quickly for templates.

HTML Documents and Cluster Initialization

The DOM defines a standard for accessing documents, like HTML and XML. 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. 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 . If a path is contained by a document di and the support of the path is at least the given minimum support threshold tdi , the path is called an essential path of di. For a web document set D with its path set PD, we use matrix ME with 0/1 values to represent the documents with their essential paths. The value at a cell in the matrix ME is 1 if a path pi is an essential path of a document dj. Otherwise, it is 0. We next illustrate the representation of a clustering of web documents. Let us assume that we have m clusters for a web document set D. A cluster ci is denoted by Ti is a set of paths representing the template of ci and Di is a set of documents belonging to ci. In our clustering model, we allow a document to be included in a single cluster only.

4. Proposed System

Clustering With MDL Cost

In this module we have clustering algorithm TEXT-MDL. The input parameter is a set of documents D, where di is the ith document. The output result is a set of clusters C, where ci is a cluster represented by the template paths Ti and the member documents Di. A clustering model C is denoted by two matrices MT and MD and the goodness measure of the clustering C is the MDL cost. TEXT-MDL is an agglomerative hierarchical clustering algorithm which starts with each input document as an individual cluster. When a pair of clusters is merged, the MDL cost of the clustering model can be reduced or increased. The procedure GetBestPair finds a pair of clusters whose reduction of the MDL cost is maximal in each step of merging and the pair is repeatedly merged until any reduction is not possible. In order to calculate the MDL cost when each possible pair of clusters is merged, the procedure GetMDLCost(ci, cj, C), where ci and cj are a pair to be merged and C is the current clustering, is called in GetBestPair and C is updated by merging the best pair of clusters. GetBestPair should recalculate the MDL cost reduction of every pair at each iteration of while loop. We introduce an approximate MDL cost model and use MinHash to significantly reduce the time complexity. Get MDL Cost with the approximate entropy model.

Estimation of MDL Cost with MinHash

We will present how we can estimate the MDL cost of a clustering by MinHash not only to reduce the dimensions of documents but also to find quickly the best pair to be merged in the MinHash signature space. The Jaccard's coefficient between two sets S1 and S2 and the Min-Wise independent permutation is a well-known Monte Carlo technique that estimates the Jaccard's coefficient by repeatedly assigning random ranks to the universal set and comparing the minimum values from the ranks of each set. Consider a set of random permutations on a universal set U and a set S1. Let P be the rank of rj in a permutation and is called minwise independent if we have for every set S1 and every S2. Then, for any sets S1, S2 MDL cost estimation by MinHash. Now we present the procedure GetHashMDLCost. Note that we estimate the MDL cost, but do not generate the template paths of each cluster. Thus, Tk of ck is initialized as the empty set. Instead of the template paths, the signature of ck is maintained to estimate the MDL cost. After finishing clustering, a postprocessing is needed to get the actual template paths. We refer to the processing as the template path generation step.

Clustering with MinHash

When we merge clusters hierarchically, we select two clusters which maximize the reduction of the MDL cost by merging them. Given a cluster ci, if a cluster cj maximizes the reduction of the MDL cost, we call cj the nearest cluster of ci. In order to efficiently find the nearest cluster of ci, we use the heuristic, we can reduce the search space to find the nearest cluster of a cluster ci. The previous search space to find the nearest cluster of ci was the same as the number of current clusters. But, the search space becomes the number of clusters whose Jaccard's coefficient with ci is maximal. The Jaccard's coefficient can be estimated with the signatures of MinHash and clusters whose Jaccard's coefficient with ci is maximal can be directly accessed in the signature space. we provide the procedures to find the best pair using MinHash. In TEXT-MDL, the GetBestPair is replaced by GetInitBestPair and the GetBestPair is replaced by GetHashBestPair. In GetInitBestPair, we first merge clusters with the same signature of MinHash. Next, for each cluster ci, we get clusters with the maximal Jaccard's coefficient estimated by the signatures of MinHash and compute the MDL cost of each pair. In GetHashBestPair, the steps are similar to those in GetInitBestPair.

Algorithms:

Algorithm 1: HTML Cluster Initialization

Step 1: Input HTML or XML documents
Step 2: Retrieve document node from every HTML page.
Step 3: Retrieve texts nodes from the HTML pages
Step4: Compute support of a path as number of documents, which contain the path
Step 5: Input minimum support threshold
Step 6: Compute clustering of web documents

Algorithm 2: Clustering MDLCost

Step 1: Input set of documents D
Step 2: Create two matrices MT and MD
Step 3: Merge pair of clusters and compute MDL cost of the clustering
Step 4: Compute GetBestPair to finds a pair of clusters
Step 5: Iteratively check for reduction by repeatedly merging

Algorithm 3: MDLCostMinHash

Step 1: Input the clustering Matrix
Step 2: Reduce dimensions of documents to find best pair to be merged
Step 3: Compute Jaccard's coefficient by repeatedly assigning random ranks
Step 4: Compute GetHashMDLCost
Step 5: Apply post processing to get the actual template paths.

Algorithm 4: ClusteringMinHash

Step 1: Select two clusters which maximize the reduction of the MDL cost
Step 2: Reduce the search space to find the nearest cluster
Step 3: Check Jaccard's coefficient is maximal
Step 4: Merge clusters with the same signature of MinHash
Step 5: Compute the MDL cost of each pair

5. Conclusion

Searching for information about people in the web is one of the most common activities of internet users. Around 30 percent of search engine queries include person names. However, retrieving information about people from web search engines can become difficult when a person has nicknames or name aliases. For example, the famous Japanese major league baseball player Hideki Matsui is often called as Godzilla on the web. A newspaper article on the baseball player might use the real name, Hideki Matsui, whereas a blogger would use the alias, Godzilla, in a blog entry. We will not be able to retrieve all the information about the baseball player, if we only use his real name. Identification of entities on the web is difficult for two fundamental reasons: first, different entities can share the same name (i.e., lexical ambiguity); second, a single entity can be designated by multiple names (i.e., referential ambiguity). For example, the lexical ambiguity considers the name Jim Clark. Aside from the two most popular namesakes, the formula-one racing champion and the founder of Netscape, at least 10 different people are listed among the top 100 results returned by Google for the name. On the other hand, referential ambiguity occurs because people use different names to refer to the same entity on the web.

6. Future Work

We propose as future work a lexical-pattern-based approach to extract aliases of a given name. We use a set of names and their aliases as training data to extract lexical patterns that describe numerous ways in which information related to aliases of a name is presented on the web. Next, we substitute the real name of the person that we are interested in finding aliases in the extracted lexical patterns, and download snippets from a web search engine. We extract a set of candidate aliases from the snippets. The candidates are ranked using various ranking scores computed using three approaches: lexical pattern frequency, co-occurrences in anchor texts, and page counts-based association measures. Moreover, we integrate the different ranking scores to construct a single ranking function using ranking support vector machines. We evaluate the proposed method using three data sets: an English personal names data set, an English location names data set, and a Japanese personal names data set. The proposed method reported high MRR and AP scores on all three data sets and outperformed numerous baselines and a previously proposed alias extraction algorithm. Discounting co-occurrences from hubs is important to filter the noise in co-occurrences in anchor texts. For this purpose, we proposed a simple and effective hub discounting measure. Moreover, the extracted aliases significantly improved recall in a relation detection task and render useful in a web search task.

Refernces

[1] A. Arasu and H. Garcia-Molina, “Extracting Structured Data from Web Pages,” Proc. ACM SIGMOD, 2003.
[2] D. Chakrabarti, R. Kumar, and K. Punera, “Page-Level Template Detection via Isotonic Smoothing,” Proc. 16th Int’l Conf. World Wide Web (WWW), 2007.
[3] J. Cho and U. Schonfeld, “Rankmass Crawler: A Crawler with High Personalized Pagerank Coverage Guarantee,” Proc. Int’l Conf. Very Large Data Bases (VLDB), 2007.
[4] V. Crescenzi, P. Merialdo, and P. Missier, “Clustering Web Pages Based on Their Structure,” Data and Knowledge Eng., vol. 54, pp. 279- 299, 2005.
[5] M. de Castro Reis, P.B. Golgher, A.S. da Silva, and A.H.F. Laender, “Automatic Web News Extraction Using Tree Edit Distance,” Proc. 13th Int’l Conf. World Wide Web (WWW), 2004.
[6] D. Gibson, K. Punera, and A. Tomkins, “The Volume and Evolution of Web Page Templates,” Proc. 14th Int’l Conf. World Wide Web (WWW), 2005.
[7] K. Lerman, L. Getoor, S. Minton, and C. Knoblock, “Using the Structure of Web Sites for Automatic Segmentation of Tables,” Proc. ACM SIGMOD, 2004.
[8] B. Long, Z. Zhang, and P.S. Yu, “Co-Clustering by Block Value Decomposition,” Proc. ACM SIGKDD, 2005.
[9] F. Pan, X. Zhang, and W. Wang, “Crd: Fast Co-Clustering on Large Data Sets Utilizing Sampling-Based Matrix Decomposition,” Proc. ACM SIGMOD, 2008.
[10] 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.
[11] Y. Zhai and B. Liu, “Web Data Extraction Based on Partial Tree Alignment,” Proc. 14th Int’l Conf. World Wide Web (WWW), 2005.
[13] 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.
[14] S. Zheng, D. Wu, R. Song, and J.-R. Wen, “Joint Optimization of Wrapper Generation and Template Detection,” Proc. ACM SIGKDD, 2007.

Biography

T.Prathibha **,** is pursuing her M.Tech in Software Engineering (Dept.of CSE) in SSJ Engineering College, vattinagulapally, hyderabad , A.P and India. Her areas of interests are data mining and knowledge Discovery, Software Engineering, software project management, Testing, Network Security, unified modeling language, and Mobile computing, DBMS.

B.Sukumar,Assistant professor of CSE Department has a profound teaching experience presently working in SSJ Engineering College, vattinagulapally village, Hyderabad ,AP, India.

His area of interest includes Networking, Mobile communications.

Arram Sriram, Assistant professor of IT Department, Anurag Group of Institutions, Ghatkesar, Ranga Reddy, A.P and India. His areas of interests are data mining and knowledge Discovery, Software Engineering, software project management, Testing, Network Security, unified modeling.