# **Template Detection for Large Scale Search Engines**

Liang Chen Department of Electronic Engineering Tsinghua University Beijing, 100084, P.R.China liang@compass.net.edu.cn

Shaozhi Ye Department of Computer Science University of California, Davis CA 95616, USA sye@udavis.edu

Xing Li Department of Electronic Engineering Tsinghua University Beijing, 100084, P.R.China xing@cernet.edu

# ABSTRACT

Templates in web sites hurt search engine retrieval performance, especially in content relevance and link analysis. Current template removal methods suffer from processing speed and scalability when dealing with large volume web pages. In this paper, we propose a novel two-stage template detection method, which combines template detection and removal with the index building process of a search engine. First, web pages are segmented into blocks and blocks are clustered according to their style features. Second, similar contents sharing the common layout style are detected during the index building process. The blocks with similar layout style and content are identified as templates and deleted. Our experiment on eight popular web sites shows that our method achieves 20-40% faster than *shingle* and *SST* methods with close accuracy.

# **Categories and Subject Descriptors**

H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval—Information filtering

# **General Terms**

Algorithms, Experimentation

#### Keywords

Template detection, web page segmentation, clustering

# 1. INTRODUCTION

Templates are common parts shared by many web pages in a web site. Typical templates include navigation bars, advertisement, privacy policy and contact information, etc. Although templates are widely used in web site design, they are negative factors for information retrieval. First, the words in template hurt relevance computation by their irrelevant content. Second, the link distribution of web pages is skewed by the duplicated links in templates. The templates problem has been demonstrated in [1, 2, 3, 4].

SAC'06 April 23-27, 2006, Dijon, France

Copyright 2006 ACM 1-59593-108-2/06/0004 ...\$5.00.

Much work has been done on template detection and removal [1, 2, 3, 4]. Although these methods achieve high accuracy, they have two major drawbacks: speed and scalability. The bottleneck for speed is to identify similar content. There are three methods to compare content: (1) compare the shingle (fingerprint of a string) of the content [1, 2]; (2) compare the keywords [3]; (3) exactly match the contents by word [4]. All these methods cost much computation overhead, making the template detection time consuming.

In this paper, we propose an approach to combine template detection and removal with index building process of a search engine. Our method is based on the following two assumptions: (1) templates in a site share a common layout style; (2) templates share similar contents. Corresponding to these two assumptions, our method consists of two stages. In the first stage, in order to capture common layout style, web pages are segmented into blocks and then clustered according to their layout styles. In the second stage, our method uses the word offset distribution to identify similar contents, which can be obtained during the index building process of a search engine. When index is being built, the word offset distributions are computed in each block. The blocks sharing close distributions are identified as similar contents. Blocks having both common layout style and similar content are identified as templates and deleted from index. This combination is promising to solve the template problem in large scale search engines. First, it costs less computation overhead to measure the similarity between contents. Second, it is scalable for large scale search engines by adding a pipeline stage into the index building process, which just costs some computation and does not compete the index builder with disk I/O.

The remainder of this paper is organized as follows. Section 2 reviews the previous work on template detection. Section 3 describes our approach and the experiment result is presented in Section 4. We conclude the paper with Section 5.

# 2. RELATED WORK

Many methods have been proposed for the template problem. [1] employs a notion of *pagelet* to segment a web page. A *pagelet* is determined by the number of hyperlinks in a HTML element. The *pagelet* whose frequency exceeds a threshold is identified as template. However, the partition only depends on the number of hyperlinks in an HTML element, and thus may not reflect the layout of a web page. Poor segmentation will cause the failure of template detection. A similar method is proposed in [2], which partitions

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

web pages based on HTML tag <TABLE>.

[3] employs the same partition method as [2]. Keywords of each block content are extracted to compute entropy for the block. Blocks with small entropy are identified as templates. The approach faces two problems: (1) its accuracy relies on feature selection. (2) it is time consuming to extract features for all the blocks.

In [4], a tree structure, site style tree (SST), is introduced to capture the common presentation style of web pages. SSTis a tree that merges all the DOM Trees of all web pages. Entropy of SST element is computed to determine which element is template. Compared with other methods, SSTachieves a higher accuracy. However, several drawbacks limit its scalability: (1) when a new DOM Tree is merged, the content of each DOM Tree's element must be compared with all the branches in SST to determine whether it is a new branch or an existed one. This process costs much computation. (2) When dealing with large volume web pages, SST can not fit into memory.

Besides the research on template detection, there are many other related studies: web segmentation [5], fragment detection [6], block-leveled search [7, 8], block-leveled link analysis [9], and content extraction [10]. Some studies propose other methods to solve similar problems, e.g. statistical method [11] and machine learning [12, 13].

#### **3. OUR APPROACH**

The primary goal of our method is to detect and remove templates from indices of search engines. Our method consists of two stages. First, web pages are segmented into blocks, and blocks with common layout style are grouped. Second, during the index building process, similar contents are detected in those blocks with common layout style and removed from the index.

In the first stage, our approach of web segmentation is similar to [2, 3]. However, [2, 3] omit the layout style information, e.g. position in the web page, background color, size. Searching template candidates in the block corpus of a web site is inefficient, because most templates only exist in the blocks sharing the same layout style. In our approach, blocks are clustered according to their layout style information after the web segmentation. Searching templates is only applied to those blocks with the common layout style.

In the second stage, we use word offset distribution in the block to measure similarity between contents. Our experiment result shows that word offset distribution provides enough information, TF(Term Frequency) and positions of each word, to represent the content and thus can be applied to measure similarity between contents. In practical, word offset distribution is denoted by offset sequence. In our work, we explore Bloom Filter technique for finding similar offset sequences. Bloom Filter costs low memory and computation overhead, which improves the speed of our method.

#### 3.1 Web Segmentation and Block Clustering

To segment a web page, HTML elements of the web page are parsed. Only the elements which determine the layout structure are used, e.g.,  $\langle TABLE \rangle$ ,  $\langle TD \rangle$ ,  $\langle TR \rangle$ ,  $\langle P \rangle$ ,  $\langle UL \rangle$ . As mentioned in [2],  $\langle TR \rangle$  and  $\langle TD \rangle$  within  $\langle TABLE \rangle$  partition the table into smaller units. Most of these smaller units only contain a single object, which cannot separate template and non-template blocks. Hence, we omit  $\langle TD \rangle \langle TR \rangle$  and use the rest as the boundary to



Figure 1: Block Tree

segment web pages.

When a web pages is segmented, layout style information is extracted. A block's layout style information includes its position in the web page and HTML elements' attributes, e.g. background color, table size. An array of integers is used to represent the HTML elements' attributes. A number sequence is used to represent the block's position, whose the *i*th number represents the block's position in the *i*th layer. Figure 1 gives an example. The dark block's position is sequence "32": it is under the 3rd block in the first layer and is the 2nd block in the second layer.

According to layout style information, blocks are clustered by Single-pass method. Each cluster is called *block-style cluster* (BSC), which is defined as follows.

Definition 1. a block-style cluster (BSC) is a set of blocks  $\phi = \{b_1, b_2, ..., b_k\}$ , which satisfies the following requirements:

(1)  $A_i = A_j$ , for all  $i \neq j$ ;

(2)  $D_i = D_j$ , for all  $i \neq j$ .

Where  $A_i$  denotes the HTML attributes of a block  $b_i$  and  $D_i$  denotes the position of  $b_i$ .

## 3.2 Template Detection during Index Building

Most modern search engines adopt inverted index. In our approach, during the process of building index, a 256K hash table is created to store words. Each word item points to a linked list whose element stores the block ID and the offset sequence in this block. When the size of the hash table reaches 256M, it is written to disk. After all the web pages are indexed, all the temporary hash tables are merged to build a whole index for the search engine.

For each word in the hash table, all the offset sequences are clustered by Single-pass method. Each cluster is called *word-feature cluster (WFC)*, which is defined as follows. Figure 2 demonstrates *WFC*'s definition.

Definition 2. a word-feature cluster (WFC) of word  $W_n$  is a set of blocks  $\phi = \{b_1, b_2, ..., b_k\}$  which satisfies the following requirements:

(1) Blocks  $b_1, b_2, ..., b_k$  belong to the same BSC;

(2)  $W_n$ 's offset sequences in blocks  $b_1, b_2, ..., b_k$  are "similar". Bloom Filter is used to determine the similarity between offset sequences, which will be described in Section 3.3.2.

Based on the definition of *WFC*, the problem of computing similarity of content is decomposed into measuring similarity of words (words' offset sequences). Hence, we introduce a notion template word.

Definition 3. word  $W_n$  in block  $b_i$  is a template word, if there is a set of blocks  $\phi = \{b_1, b_2, ..., b_k\}, b_i \in \phi$  which



Figure 2: BSC and WFC in Hash Table

follows the rules:

(1)  $b_1, b_2, ..., b_k$  belong to the same WFC of  $W_n$ ; (2)  $|\phi| \ge 5$ 

If the ratio of template words to the whole words in a block reaches a threshold (*twr-theshold*), the block is identified as template and all the words in this block are removed from hash table. *tb-theshold* is set to be 0.8 in our experiment. Besides *tb-theshold*, we also define non-template word ratio threshold (*ntwr-theshold*). According to *tb-theshold*, *ntwr-theshold* is set to be 0.2. If the non-template word ratio exceeds *ntwr-theshold*, this block is identified as a non-template block. Thus there is no necessary to compare the rest words, which reduces the time consumption.

## 3.3 Implementation

Besides the basic method, there are some implementation issues.

#### 3.3.1 Difference Offset

. In our approach, difference offset is applied instead of absolute offset for offset sequence. The advantage of difference offset is that when a small fragment in the context changes, only offsets within the fragment change. A majority of the offset sequence remains the former value.

#### 3.3.2 Bloom Filter Like Similarity Computation

To evaluate similarity of offset sequences, we introduce bit-wise AND, which originally comes from Bloom Filter [14, 15]. It is implemented as an array of n bits. We employ four integer variables  $m_i$ , i = 1, 2, 3, 4 (128 bits totally). An offset sequence is partitioned into four equal-length segments, with each corresponding to one integer variable. Each offset in a segment maps to one bit in the corresponding variable: suppose an offset value is i. Then the *i*th bit in the array is set to 1. If the offset value exceeds 32 (an integer variable has 32 bits), the offset is divided by 32 and the remainder is mapped to one bit. If a bit is already set, it stays 1. To evaluate similarity, we compare the array of bits of one offset sequence with that of the other. In case the two arrays share a large number of 1's (bit-wise AND), they are marked similar. This evaluation method is fast since matching is only a bit-wise AND operation. Figure 3 gives an example of similarity testing.

It must be explained that Bloom Filter is a fast approximate comparison and some errors (collisions) may occur. Nevertheless, our method to detect similar content is based on most words in a block. Thus this disadvantage can be re-



Figure 3: 75% offset matches. In this paper, offset sequence similarity threshold is set to be 80%

duced through average. Our experimental results, presented in the Section 4, show that the error ratio is acceptable.

#### 3.3.3 Skip low DF words

When the hash table is scanned to detect template word, there is no necessary to scan all the words. Since templates occur in many web pages, DF (Document Frequency) of a template word must exceed a certain threshold, thus we can skip the words with low DF. In fact, the words with low DF contribute a big proportion to the whole index due to Zip's law. Our statistics on some sites (listed in Table 1 shows that approximate 50% words' DF is lower than 3.

# 4. EXPERIMENT

### 4.1 Datasets

Since our method aims at search engines, which deal with various types of web sites, the sites we choose to experiment should be representative. We choose several popular sites with different categories, including commercial sites (e.g. www.pcmag.com), homepage of a company (e.g. www.sun.com), news site (e.g. news.sohu.com), and education sites (e.g. www.edu.cn). The last four sites have Chinese web pages. The testing sites are listed in Table 1.

#### 4.2 Query Accuracy

Query accuracy is the most important goal to solve the template problem for search engines. Here we use the dataset from www.tsinghua.edu.cn, for we have the query log from the site search engine for two years. In the experiment, the top 20 popular queries extracted from the log are used as our experiment queries. We ask 20 people to label the relevant web pages in top 10 results manually. There are two kinds of "relevant pages": (1) the pages a user expects to retrieve;

	•		
Web Sites	The Number of Web Pages		
www.sun.com	4,546		
www.pcmag.com	11,118		
www.cnn.com	7,883		
www.nba.com	10,886		
news.sohu.com	5,027		
www.edu.cn	23,738		
www.tsinghua.edu.cn	14,393		
www.bupt.edu.cn	27,876		

Table 1: Web Sites for Experiment



Figure 4: Number of Relevant Results in Retrieved Web Pages. Average number of relevant results are 6.5, 6.85 and 4.65, corresponding to "Our Method", "shingle" and "No Template Detection" respectively.

(2) although some pages are not expected, they are informative and attract user's attention. The evaluation results are shown in Figure 4.

From the Figure 4, we can see that template detection and removal has a notable effect on improving retrieved results' relevancy. Templates' influence on query accuracy can be divided into two categories: query appears in the template block or not. In the first category, a lot of web pages with irrelevant main contents will be retrieved. In the second category, although main contents of retrieved pages contain the query, relevancy of web pages will be decreased by templates. Two typical ranking function is listed below to demonstrate it.

#### Okapi BM25

$$\sum_{T \in Q} \frac{3 \times tf}{0.5 + 1.5 \times \frac{length}{length_{avg}} + tf} \times \log \frac{N - df + 0.5}{df + 0.5} \times tf$$

#### Pivoted TFIDF

$$\sum_{T \in Q} \frac{1 + \log\left(tf\right)}{1 + \log(tf_{avg})} \times \log\frac{N+1}{df} \times \frac{1}{0.8 + 0.2 \times \frac{length}{length_{avg}}} \times tf$$

In these two functions, there is a common term  $\frac{length}{length_{avg}}$ . Templates increase length of a web page and thus the denominators of ranking functions, which finally decreases the web page's relevancy.

#### 4.3 Template Detection Accuracy

In this section, we compare the accuracy of our method

 Table 2: Template Detection Accuracy

Web Sites	Our Method	shingle	SST
www.sun.com	93	96	89
www.pcmag.com	94	92	99
www.cnn.com	92	94	98
www.nba.com	89	90	92
news.sohu.com	95	97	100
www.edu.cn	94	95	99
www.tsinghua.edu.cn	91	92	79
www.bupt.edu.cn	93	92	80



Figure 5: Different Templates in Tsinghua

with shingle [2] and SST [4]. As mentioned in section 2, SST can only deal with a small number of web pages, for it stores all the DOM Trees in memory. For large sites, e.g. the sites listed in Table 1, we can not apply SST directly. In [4], this problem is avoided by sampling only about 500 web pages in a site. In our experiment, we apply SST to large sites by multiple samples.

We randomly sample 100 web pages from each site. People are asked to identify if the templates in a web page are recognized correctly. A score is assigned to each page. The score is computed as follows:

$$score = \frac{number \ of \ right-detect \ template \ blocks}{number \ of \ total \ actual \ template \ blocks}$$

We summarize the result with Table 2.

From Table 2, we could see that *SST*'s accuracy is poor in www.tsinghua.edu.cn and www.bupt.edu.cn, which is caused by sampling. In *SST*, only 500 pages are sampled for template detection. This works in many commercial sites, for these sites have only a few templates and these templates are similar. Sampling a small amount of web pages does not hurt the accuracy, as the results in www.pcmag.com and news.sohu.com. However, when dealing with a web site having multiple templates, its accuracy relies on sampling and is unstable.

Web Sites	Our Method	shingle	SST
www.sun.com	10.70	16.52	18.38
www.pcmag.com	16.75	19.92	23.09
www.cnn.com	14.43	18.74	20.54
www.nba.com	14.10	17.40	19.55
news.sohu.com	17.32	21.32	24.06
www.edu.cn	16.68	19.90	22.89
www.tsinghua.edu.cn	16.50	20.76	22.13
www.bupt.edu.cn	17.13	20.13	24.13
Average time	15.45	19.34	21.85
Saving Time		20.09%	41.39%

Table 3: Time Consumption

Figure 5 shows different templates in www.tsinghua.edu.cn. In this case, especially when the web site has many pages, random sampling cannot guarantee the accuracy based on a small sampled set.

## 4.4 Time Consumption

Time consumption is an important concern in this paper for we target on deployment in large scale search engines. In order to compare with SST, we sample 500 web pages from each site as [4]. Our method involves both parsing and indexing, while others only involve in parsing. In order to be fair, when we run the other methods, we also build index hash table. The result is shown in Table 4. As shown in Table 3, our method saves 20.09% compared with *shingle*, and 41.39% compared with *SST*. The main reason is that the word offset sequence which we use to compute similarity is available during index building process. It costs much less computation than the other two methods.

#### 4.5 Discussion

Results in Table 2 and Table 3 show several limitations of existed methods (*shingle* and *SST*) when they are applied in large scale search engines. For *shingle*, because it omits the layout style information, time for searching similar content in all shingle values increases greatly with the increase of web pages. Time for computing shingle values is another overhead. For *SST*, it is ineffective for large scale sites for its unstable accuracy after sampling and low speed. Compared with these two methods, our approach achieves a stable high accuracy and fast speed.

Furthermore, combination with index building makes it easy to keep up with the increase of index volume. And since there is no extra I/O operation in our method, we can add a pipeline stage to the index building process for template detection, which increases indexing computation overhead but does not compete with indexer with disk I/O operation. Disk I/O is the most time consuming stage in index building, thus we can reduce the overall execution time by applying template detection when the indexer writes data to disk.

## 5. CONCLUSION

In this paper, we propose to combine template detection and removal with the index building process in large scale search engines. To capture common layout style, we employ web segmentation and block style cluster. To capture similar content, we use word offset sequences, which is available without extra computation effort during index building process. Through these two steps, template are detected and deleted from index. Our experiment on eight popular large web sites indicates that our method is promising for large scale search engines by high accuracy and fast speed.

## 6. **REFERENCES**

- Ziv Bar-Yossef, Sridhar Rajagopalan. Template Detection via Data Mining and its Applications. In Proc. of the WWW'02 Conf., pages 580–591, 2002
- [2] Ling Ma, Nali Goharian, Abdur Chowdhury. Extracting unstructured Data from Template Generated Web Document. In Proc. of the CIKM'03 Conf., pages 516–519, 2003
- [3] Shian-Hua Lin, Jan-Ming Ho. Discovering Informative Content Blocks from Web Documents. In Proc. of the SIGKDD'02 Conf., pages 588–593, 2002
- [4] Lan Yi, Bing Liu, Xiaoli Li. Eliminating Noisy Information in Web Pages for Data Mining. In Proc. of the SIGKDD'03 Conf., pages 296–305, 2003
- [5] Deng Cai, Shipeng Yu, Ji-rong Wen and Wei-Ying Ma. Extracting Content Structure for Web Pages based on Visual Representation. In Proc. of the APWeb'03 Conf., number 2642 in LNCS, pages 406–417, 2003
- [6] Lakshmish Ramaswamy, Arun Lyengar, Ling Liu, Fre Douglis. Automatic Detection of Fragments in Dynamically Generated Web Pages. In Proc. of the WWW'04 Conf., pages 443–454, 2004
- [7] Xiaoli Li, Tong-Heng Phang, Mingqing Hu, Bing Liu. Using micro information units for internet search. In Proc. of the CIKM'02 Conf., pages 566-573, 2002
- [8] Deng Cai, Shipeng Yu, Ji-Rong Wen, Wei-Ying Ma. Block-based Web Search. In Proc. of the SIGIR'04 Conf., pages 456-463, 2004
- [9] Deng Cai, Xiaofei he, Ji-Rong Wen, Wei-Ying Ma. Block-level Link Analysis. In Proc. of the SIGIR'04 Conf., pages 440-447, 2004
- [10] Davi de Castro Reis, Paulo B. Golgher, Altigran S. da Silva, Alberto H.F.Laender. Automatic Web News Extraction Using Tree Edit Distance. In Proc. of the WWW'04 Conf., pages 502-511, 2004
- [11] Arvind Arasu, Hector Garcia-Molina. Extracting structured data from Web pages. In Proc. of the SIGMOD'03 Conf., pages 337-348, 2003
- [12] Davision, B.D. Recognizing Nepotistic links on the Web. In Proc. of the AAAI Conf., pages 23-28, 2000
- [13] Nicholas Kushmerick. Learning to remove Internet advertisements. In Proc. of the third annual conference on Autonomous, Agents, 1999
- [14] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. In Commun. ACM, 13(7):422-426, 1970
- [15] Navendu jain, Mike Dahlin, Renu Tewari. Using Bloom Filters to Refine Web Search Results. In the Eighth Workshop on Web and Database (WebDB'05), 2005