Extracting Content Structure for Web Pages based on Visual Representation

Deng Cai[1*], Shipeng Yu [2*], Ji-Rong Wen[*] and Wei-Ying Ma[*]

[*] Microsoft Research Asia
{jrwen, wyma}@microsoft.com
[1] Tsinghua University, Beijing, P.R.China
caideng00@mails.tsinghua.edu.cn
[2] Peking University, Beijing, P.R.China
ysp@is.pku.edu.cn

Abstract. A new web content structure based on visual representation is proposed in this paper. Many web applications such as information retrieval, information extraction and automatic page adaptation can benefit from this structure. This paper presents an automatic top-down, tag-tree independent approach to detect web content structure. It simulates how a user understands web layout structure based on his visual perception. Comparing to other existing techniques, our approach is independent to underlying documentation representation such as HTML and works well even when the HTML structure is far different from layout structure. Experiments show satisfactory results.

1 Introduction

Today the Web has become the largest information source for people. Most information retrieval systems on the Web consider web pages as the smallest and undividable units, but a web page as a whole may not be appropriate to represent a single semantic. A web page usually contains various contents such as navigation, decoration, interaction and contact information, which are not related to the topic of the web page. Furthermore, a web page often contains multiple topics that are not necessarily relevant to each other. Therefore, detecting the content structure of a web page could potentially improve the performance of web information retrieval.

Many web applications can utilize the content structures of web pages. For example, some researchers have been trying to use database techniques and build wrappers for web documents [3]. If a web page can be divided into semantic related parts, wrappers can be more easily matched and data can be more likely extracted. Link structure analysis can also make good use of the content structures of web pages. Links at different parts of a page usually act as different functions and contribute to the PageRank [2] or HITS [11] differently. Recent works on topic distillation [4] and focused crawling [5] show the usefulness of page segmentation on information analysis. Furthermore, adaptive content delivery on small handheld devices also requires the detection of underlying content structure of a web page to facilitate the browsing of a large page by partitioning it into smaller units [10].

People view a web page through a web browser and get a 2-D presentation which has many visual cues to help distinguish different parts of the page. Generally, a web page designer would organize the content of a web page to make it easy for reading. Thus, semantically related content is usually grouped together and the entire page is divided into regions for different contents using explicit or implicit visual separators such as lines, blank areas, images, font sizes, colors, etc [16]. This motivates our work to segment a web page into semantically related content blocks from its visual presentation. If we can reconstruct the structure of a page corresponding to human visual perception, it will better reflect the semantic structure. In this paper, we propose VIPS (**VI**sion-based **P**age **S**egmentation) algorithm to extract the content structure for a web page. The algorithm makes full use of page layout features and tries to partition the page at the semantic level. Each node in the extracted content structure will correspond to a block of coherent content in the original page.

The paper is organized as follows. Section 2 presents the related works. Section 3 defines the web page content structure based on vision. The details of the VIPS algorithm are introduced in Section 4. The experimental results are reported in Section 5. Section 6 summarizes our contributions and concludes the paper.

2 Related Works

The applications mentioned in the introduction indicate the need of techniques for extracting the content structure of a web page. Tag tree or Document Object Model (http://www.w3.org/DOM/) provides each web page a fine-grained structure, illustrating not only the content but also the presentation of the page. Many researchers [10, 12, 15] have considered using the tag information and dividing the page based on the type of the tags. Useful tags include <P> (for paragraph), <TABLE> (for table), <UL> (for list), <H1>~<H6> (for heading), etc.

Some other algorithms also consider the content or link information besides the tag tree. Embley [8] use some heuristic rules to discover record boundaries within a page, which assist data extraction from the web page. Chakrabarti [4] addresses the fine-grained topic distillation and dis-aggregates hubs into regions by analyzing link structure as well as intra-page text distribution.

A Function-based Object Model (FOM) of a web page is proposed by Chen [6] for content understanding and adaptation. Every undividable element in the tag tree is called a basic object and can be grouped into a composite object. A function type can be defined to each object and helps to build a hierarchical structure for the page. However, the grouping rules and the functions are hard to define accurately, and thus make the whole tree-constructing process very inflexible.

In [16] and [6], some visual cues are used in DOM analysis. They try to identify the logic relationships within the web content based on visual layout information, but these approaches still rely too much on the DOM structure. Gu [9] tries to construct a web content structure by breaking out the DOM tree and comparing similarity among all the basic DOM nodes. Since a normal web page may have hundreds of basic elements, the algorithm is time-consuming and inflexible, not capable to deal with a large amount of web pages.

3 Vision-based Content Structure for Web Pages

Similar to [6], we define the *basic object* as the leaf node in the DOM tree that cannot be decomposed any more. Although a DOM structure provides a hierarchy for the basic objects in a web page, it is more for representation rather than content organization. In this paper, we propose the *vision-based content structure*, where every node, called a *layout block*, is a basic object or a group of basic objects. The nodes in the content structure do not necessarily have a mapping to the nodes in the DOM tree.

Similar to the description of document representation in [14], the basic model of vision-based content structure for web pages is described as follows.

A web page Ω is specified by a triple $\Omega = (O, \Phi, \delta)$. $O = \{\Omega^1, \Omega^2, ..., \Omega^N\}$ is a finite set of objects or sub-web-pages. All these objects are not overlapped. Each object can be recursively viewed as a sub-web-page and has a subsidiary content structure. $\Phi = \{\varphi^1, \varphi^2, ..., \varphi^T\}$ is a finite set of visual separators, including horizontal separators and vertical separators. Every separator has a weight indicating its visibility, and all the separators in the same Φ have same weight. δ is the relationship of every two blocks in O and can be expressed as: $\delta = O \times O \to \Phi \cup \{NULL\}$. Suppose Ω_i and Ω_j are two objects in O, $\delta(\Omega_i, \Omega_j) \neq NULL$ indicates that Ω_i and Ω_j are exactly separated by the separator $\delta(\Omega_i, \Omega_j)$ or we can say the two objects are adjacent to each other, otherwise there are other objects between the two blocks Ω_i and Ω_j.

Since each Ω_i is a sub-web-page of the original page, it has similar content structure as Ω. Recursively, we have $\Omega_s^t = (O_s^t, \Phi_s^t, \delta_s^t)$, $O_s^t = \{\Omega_{st}^1, \Omega_{st}^2, ..., \Omega_{st}^{N_{st}}\}$, $\Phi_s^t = \{\varphi_{st}^1, \varphi_{st}^2, ..., \varphi_{st}^{T_{st}}\}$ and $\delta_s^t = O_s^t \times O_s^t \to \Phi_s^t \cup \{NULL\}$ where Ω_s^t is the t^{th} object in the sub-web-page level s, N_{st} and T_{st} are the number of objects in O_s^t and number of separators in Φ_s^t.

Fig. 1 shows an example of visual-based content structure for Yahoo! Auctions page. It illustrates the layout structure and the vision-based content structure of the page. In the first level, the original web page has four objects or visual blocks VB1~VB4 and three separators $\varphi^1 \sim \varphi^3$, as specified in Fig. 1(d). Then we can further construct sub content structure for each sub web page. For example, VB2 has three offspring objects and two separators. It can be further analyzed like Fig. 1(e).

For each visual block, the *Degree of Coherence* (DoC) is defined to measure how coherent it is. DoC has the following properties:

- Ranges from 0 to 1;
- The greater the DoC value, the more consistent the content within the block;
- In the hierarchy tree, the DoC of the child is not smaller than its parent's.

We can pre-define the *Permitted Degree of Coherence* (PDoC) to achieve different granularities of content structure for different applications. The smaller the PDoC is, the coarser the content structure would be. For example in Fig. 1(a), the visual block VB2_1 may not be further partitioned with an appropriate PDoC.

The vision-based content structure is more likely to provide a semantic partitioning of the page. Every node, especially the leaf node, is more likely to convey a semantic meaning for building a higher semantic via the hierarchy. For instance, in Fig. 1(a) we can say that VB2_1_1 denotes the category links of Yahoo! Shopping auctions, and that VB2_2_1 and VB2_2_2 show details for two different comics.

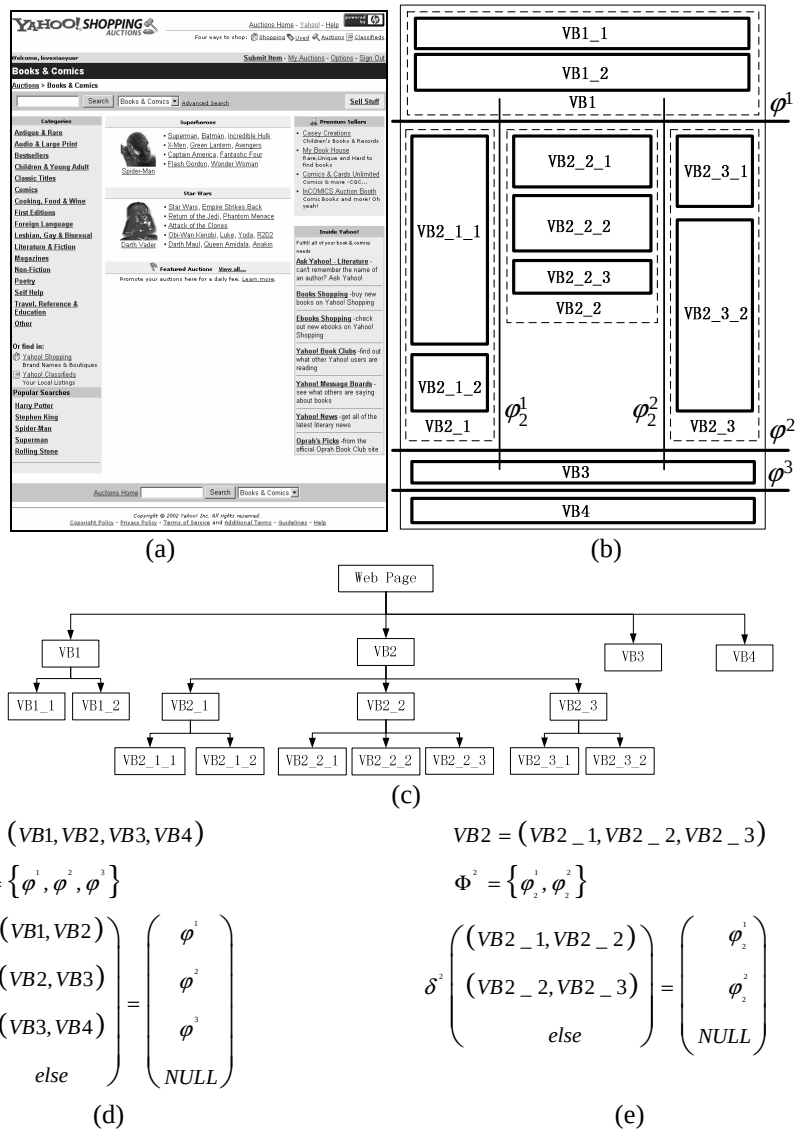

Fig. 1. The layout structure and vision-based content structure of an example page. (d) and (e) show the corresponding specification of vision-based content structure.

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. The segmentation process is illustrated in Fig. 2. 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.

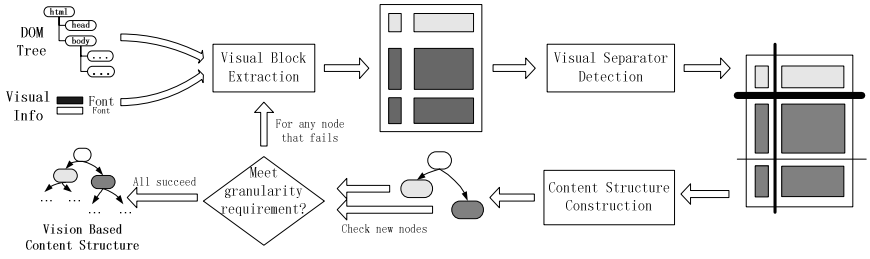

Fig. 2. The vision-based page segmentation algorithm

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.

The visual block extraction algorithm DivideDomtree is illustrated in Fig. 3. 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.

```
Algorithm DivideDomtree(pRoot, nLevel)
{
  IF (Dividable(pRoot, nLevel) == TRUE){
    FOR EACH child OF pRoot {
        DivideDomtree(child, nLevel);
    }
  } ELSE {
    Put the sub-tree (pRoot) into the
      pool as a block;
  }
}
```

(a)

```
Algorithm Dividable(pRoot, nLevel)
{
  IF (pRoot is the Top Block){
    RETURN TRUE;
  } ELSE {
    Special routines for TABLE, TR,
      TBODY, TD, P, UL, FORM;
    Heuristic rules for general tags;
  }
}
```

(b)

Fig. 3. The visual block extraction algorithm

- Size cue: We prefer to divide a DOM node if the standard deviation of size of its children is larger than a threshold.

The Tag cue and Color cue are relatively straightforward to produce corresponding heuristic rules. Below we give the details of how to use text and size cues to generate heuristic rules for block extraction. First, several definitions are given:

1. *Valid Node*: a node that can be seen through the browser. The node's width and height are not equal to zero.
2. *Block Node*: the node with tag not <A>, <B>, <FONT>, <HR>, <I>, <P>, <STRONG>, <TEXT>.
3. *Text Node*: the DOM node that only contains free text.
4. *Virtual Text Node*: The node that is not a block node and only have text node as children.

According to the above definitions, we use the following text cue and size cue based heuristic rules to further enhance the block extraction process:

- A node will be dropped if it has no valid child.
- If a node only has one valid child and this child is not Text node, then trace into the child.
- If all the children of a node are Text nodes or Virtual Text nodes, then set the DoC 1.
- If the node's size is 3 times greater than all his children's total size, divide it.
- If the node has Text node child or Virtual Text node child and the node's width or height is smaller than a threshold, set the DoC 0.8.
- Split the node which has more than two successive
 children. (It means there are many space in the middle, may be two different topics)

In addition, some tags, such as <TABLE>, <TBODY>, <TR>, <TD>, <P>, <UL> and <LI>, are very important and common in web page and are more likely to form a content coherent sub-tree. So we define some special routines in our algorithm to handle these tags and set higher thresholds for these tags in the above rules.

Take Fig. 1 as an example. VB1, VB2_1, VB2_2, VB2_3, VB3 and VB4 will be extracted at the first round. The detailed process is as follows. In Fig. 4, a DOM tree structure of VB2 is shown. In the block extraction process, when the <TABLE> node is met, it has only one valid child <TR>. We trace into the <TR> node according to the heuristic rules. The <TR> node has five <TD> children and only three of them are

valid. The first child's background color is different from its parent's background color, so the <TR> node is split and the first <TD> node is not divided further in this round and is put into the pool as a block. The second and fourth child of <TR> node is not valid and will be dropped. When the third and fifth children of <TR> are considered, we use another rule to improve the efficiency – because the first <TD> is not divided, so there will be no horizontal separators in this round projection. It is unnecessary to divide the third and fifth <TD> and they should be taken as a block in this level.

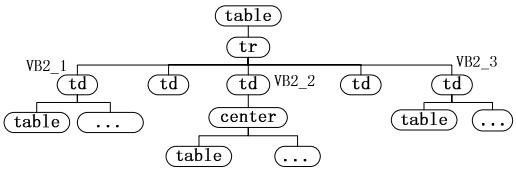

Fig. 4. DOM tree structure of VB2 in the sample page

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: (P_s, P_e), where P_s is the start pixel and P_e is the end pixel. The width of the separator is calculated by the difference between these two values.

Separator Detection. The visual separator detection algorithm is described as follows:

1. Initialize the separator list. The list starts with only one separator (P_{be}, P_{ee}) whose start pixel and end pixel are corresponding to the borders of the pool.
2. For every block in the pool, the relation of the block with each separator is evaluated
   - If the block is contained in the separator, split the separator;
   - If the block crosses with the separator, update the separator's parameters;
   - If the block covers the separator, remove the separator.
3. Remove the four separators that stand at the border of the pool.

Take Fig. 5(a) as an example in which the black blocks represent the visual blocks in the page. For simplicity we only show the process to detect the horizontal separators. At first we have only one separator that is the whole pool. As shown in Fig. 5(b), when we put the first block into the pool, it splits the separator into S1 and S2. It is same with the second and third block. When the fourth block is put into the pool, it crosses the separator S2 and covers the separator S3, the parameter of S2 is updated and S3 is removed. At the end of this process, the two separators S1 and S3 that stand at the border of the pool are removed.

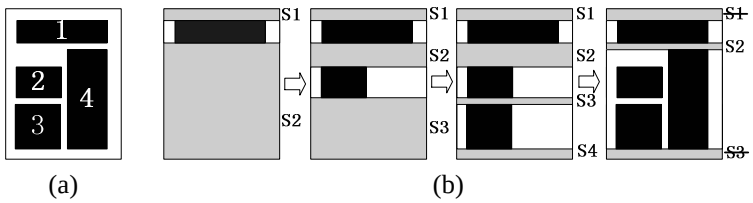

Fig. 5. A sample page and the separator detection process

Setting Weights for Separators. The separators are used to distinguish blocks with different semantics, so the weight of a separator can be assigned based on the information difference between its neighboring blocks. The following rules are used to set a weight to each separator:

- The more the distance between blocks on different side of the separator, the higher the weight.
- If a visual separator is at the same position as some tags such as <HR>, its weight is made higher.
- If the differences of font properties such as font size and font weight are more clearly on two sides of the separator, the weight will be increased. More over, the weight will be increased if font size before the separator is smaller than that after the separator.
- If background colors are different on two sides of the separator, the weight will be increased.
- When the structures of the blocks beside the separator are very similar (e.g. both are text), the weight of the separator will be decreased.

Take the third <TD> node in Fig. 4 as an example. The sub-page corresponding to this node is shown in Fig. 6(b) and the DOM tree structure is shown in Fig. 6(a). We can see that many nodes in the DOM tree are invalid in our definition and cannot be seen on the page. They are ignored in the block extraction process. After the block extraction phase, six blocks are put in a pool and five horizontal separators are detected. Then the weights of these separators are set based on the rules we described. In this example, the separator between Block 2 and Block 3 will get higher weight than the separator between Block 1 and Block 2 because of the different font weights. For the same reason, the separator between Block 4 and Block 5 will also get a high weight. The final separators and weights are shown in Fig. 6(c), in which a thicker line means a higher weight.

4.3 Content Structure Construction

When separators are detected and separators' weights are set, the content structure can be constructed accordingly. The construction process starts from the separators with the lowest weight and the blocks beside these separators are merged to form new virtual blocks. This process iterates till separators with maximum weights are met. The DoC of each new block is also set via similar methods described in Section 4.1.

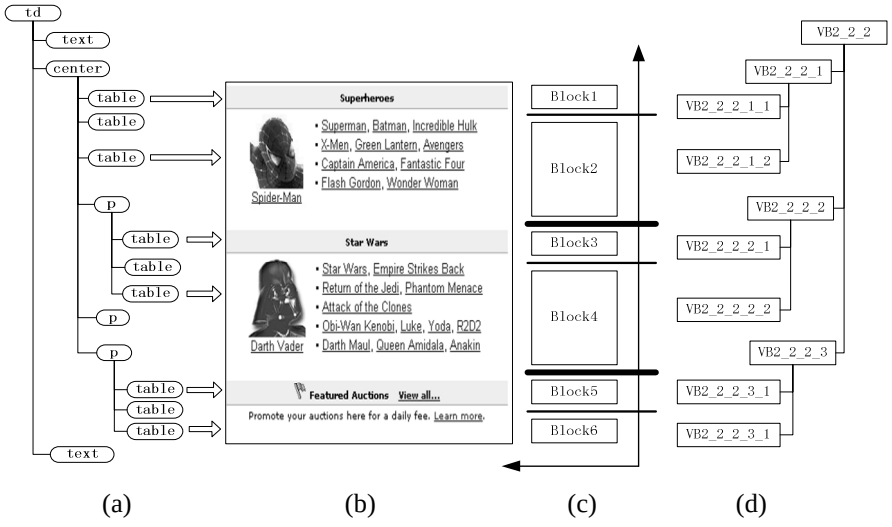

Fig. 6. Separators and weights among blocks

After that, each leaf node is checked whether it meets the granularity requirement. For every node that fails, we go to the Visual Block Extraction phase again to further construct the sub content structure within that node. If all the nodes meet the requirement, the iterative process is then stopped and the vision-based content structure for the whole page is obtained. The common requirement for DoC is that DoC > PDoC, if PDoC is pre-defined.

Take Fig. 6 as an example. In the first iteration, the first, third and fifth separators are chosen and Block 1 and 2 are merged to form the new block VB2_2_2_1. Similar merging is conducted for Block 3 and 4 (resulting a new block VB2_2_2_2) and Block 5 and 6 (resulting a new block VB2_2_2_3). The new blocks VB2_2_2_1, VB2_2_2_2 and VB2_2_2_3 are the children of VB2_2_2 and can also be viewed as a partition of VB2_2_2. Every leaf node, such as VB2_2_2_1_1, VB2_2_2_1_2 and VB2_2_2_2_1, will be checked to see whether it meets the granularity requirement. After several iterations, the final vision-based content structure of the page is constructed.

In summary, the proposed VIPS algorithm takes advantage of visual cues to obtain the vision-based content structure of a web page and thus successfully bridges the gap between the DOM structure and the semantic structure. The page is partitioned based on visual separators and structured as a hierarchy closely related to how a user would browse the page. Content related parts could be grouped together even if they are in different branches of the DOM tree.

VIPS is also very efficient. Since we trace down the DOM structure for visual block extraction and do not analyze every basic DOM node, the algorithm is totally top-down. Furthermore, the PDoC can be pre-defined, which brings significant flexibility to segmentation and greatly improve the performance.

5 Experiments

We provide some performance evaluation of our proposed VIPS algorithm based on a large collection of web pages from Yahoo. We also conduct experiments to evaluate how the algorithm can be used to enhance information retrieval on the Web.

5.1 Performance of VIPS algorithm

We selected 140 web pages from popular sites listed in 14 main category of Yahoo directory (http://www.yahoo.com). The web content structure detection algorithm is run against these pages and the results are assessed by human judgments. Table 1 shows the result.

Table 1. Evaluation of the quality of page analysis

Human judgment	Number of pages
Perfect	86
Satisfactory	50
Failed	4

As can be seen, 86+50=136 (97%) pages have their content structures correctly detected. For those "failed" cases, one major reason is that the browser (i.e. Internet Explorer in our experiments) provides wrong position information so that our algorithm cannot get the correct content structure. Another reason lies in that several pages use images (e.g., a very thin image that represents a line) to divide different content blocks. Our algorithm currently does not handle this situation.

5.2 Experiments on web information retrieval

Query expansion is an efficient way to improve the performance of information retrieval [7]. The quality of expansion terms is heavily affected by the top-ranked documents. Noise and multi-topics are two major negative factors for expansion term selection in the web context. Since our VIPS algorithm can group semantically related content into a segment, the term correlations within a segment will be much higher than those in other parts of a web page. With improved term correlations, high-quality expansion terms can be extracted from segments and used to improve information retrieval performance.

We choose Okapi [13] as the retrieval system and WT10g [1] in TREC-9 and TREC 2001 Web Tracks as the data set. WT10g contains 1.69 million pages and amounts to about 10G. We use the 50 queries from TREC 2001 Web Track as the query set and only the TOPIC field for retrieval, and use Okapi's BM2500 as the weight function and set $k_1 = 1.2$, $k_3 = 1000$, $b = 0.75$, and $avdl = 61200$.

An initial list of ranked web pages is obtained by using any traditional information retrieval methods. Then we apply our page analysis algorithm with a PDoC 0.6 to the

top 80 pages and get the set of candidate segments. The most relevant (e.g. top 20) segments from this candidate set are used to select expansion terms. These selected terms are used to construct a new expanded query to retrieve the final results.

We compared our method with the traditional pseudo-relevance feedback algorithm. Note that our method selects expansion terms from blocks while traditional methods select expansion terms from entire web pages. The experimental result is shown in Fig. 7.

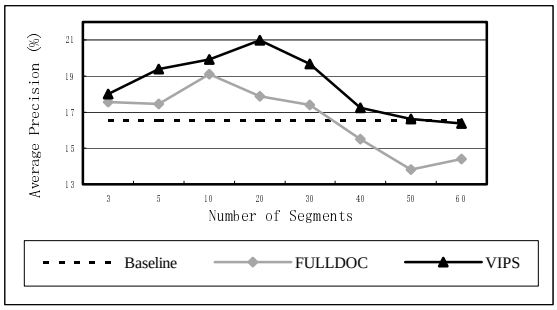

Fig. 7. Performance comparison of pseudo-relevance feedback based on two different ways of selecting query expansion terms. VIPS is our method which selects query expansion terms from blocks while FULLDOC represents traditional approaches where the entire pages are used for expansion term selection.

From Fig. 7, we can see that pseudo-relevance feedback based on web page content structure significantly outperforms traditional methods and achieves about 27% performance improvement on the Web Track dataset (refer to [17] for more details). The experiments clearly show that web page content structure is very helpful to detect and filter out noisy and irrelevant information. Thus better expansion terms can be selected to improve retrieval performance.

6 Conclusion

In this paper a new approach of extracting web content structure based on visual representation was proposed. The produced web content structure is very helpful for applications such as web adaptation, information retrieval and information extraction. By identifying the logic relationship of web content based on visual layout information, web content structure can effectively represent the semantic structure of the web page. An automatic top-down, tag-tree independent and scalable algorithm to detect web content structure was presented. It simulates how a user understands the layout structure of a web page based on its visual representation. Compared with traditional DOM based segmentation method, our scheme utilizes useful visual cues to obtain a better partition of a page at the semantic level. It also is independent of physical realization and works well even when the physical structure is far different from visual presentation. The experimental result has shown that our proposed scheme achieves

very satisfactory performance. We plan to apply the scheme for adaptive content delivery to facilitate better web browsing on mobile devices for our future works.

References

1. Bailey, P., Craswell, N., and Hawking, D., Engineering a multi-purpose test collection for Web retrieval experiments, Information Processing and Management, 2001.
2. Brin, S. and Page, L., The Anatomy of a Large-Scale Hypertextual Web Search Engine, In the Seventh International World Wide Web Conference, Brisbane,Australia, 1998.
3. Buneman, P., Davidson, S., Fernandez, M., and Suciu, D., Adding Structure to Unstructured Data, In Proceedings of the 6th International Conference on Database Theory (ICDT'97), 1997, pp. 336-350.
4. Chakrabarti, S., Integrating the Document Object Model with hyperlinks for enhanced topic distillation and information extraction, In the 10th International World Wide Web Conference, 2001.
5. Chakrabarti, S., Punera, K., and Subramanyam, M., Accelerated focused crawling through online relevance feedback, In Proceedings of the eleventh international conference on World Wide Web (WWW2002), 2002, pp. 148-159.
6. Chen, J., Zhou, B., Shi, J., Zhang, H., and Wu, Q., Function-Based Object Model Towards Website Adaptation, In the 10th International World Wide Web Conference, 2001.
7. Efthimiadis, N. E., Query Expansion, In Annual Review of Information Systems and Technology, Vol. 31, 1996, pp. 121-187.
8. Embley, D. W., Jiang, Y., and Ng, Y.-K., Record-boundary discovery in Web documents, In Proceedings of the 1999 ACM SIGMOD international conference on Management of data, Philadelphia PA, 1999, pp. 467-478.
9. Gu, X., Chen, J., Ma, W.-Y., and Chen, G., Visual Based Content Understanding towards Web Adaptation, In Second International Conference on Adaptive Hypermedia and Adaptive Web-based Systems (AH2002), Spain, 2002, pp. 29-31.
10. Kaasinen, E., Aaltonen, M., Kolari, J., Melakoski, S., and Laakko, T., Two Approaches to Bringing Internet Services to WAP Devices, In Proceedings of 9th International World-Wide Web Conference, 2000, pp. 231-246.
11. Kleinberg, J., Authoritative sources in a hyperlinked environment, In Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms, 1998, pp. 668-677.
12. Lin, S.-H. and Ho, J.-M., Discovering Informative Content Blocks from Web Documents, In Proceedings of ACM SIGKDD'02, 2002.
13. Robertson, S. E., Overview of the okapi projects, Journal of Documentation, Vol. 53, No. 1, 1997, pp. 3-7.
14. Tang, Y. Y., Cheriet, M., Liu, J., Said, J. N., and Suen, C. Y., Document Analysis and Recognition by Computers, Handbook of Pattern Recognition and Computer Vision, edited by C. H. Chen, L. F. Pau, and P. S. P. Wang World Scientific Publishing Company, 1999.
15. Wong, W. and Fu, A. W., Finding Structure and Characteristics of Web Documents for Classification, In ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD), Dallas, TX., USA, 2000.
16. Yang, Y. and Zhang, H., HTML Page Analysis Based on Visual Cues, In 6th International Conference on Document Analysis and Recognition, Seattle, Washington, USA, 2001.
17. Yu, S., Cai, D., Wen, J.-R., and Ma, W.-Y., Improving Pseudo-Relevance Feedback in Web Information Retrieval Using Web Page Segmentation, To appear in the Twelfth International World Wide Web Conference (WWW2003), 2003.