Homepage Live: Automatic Block Tracing for Web Personalization*

Jie Han[1] Dingyi Han[1] Chenxi Lin[2] Hua-Jun Zeng[2] Zheng Chen[2] Yong Yu[1]

[1]Dept. of Computer Science and Engineering
Shanghai Jiao-Tong University
Shanghai 200030, P. R. China
{micro_j, handy, yyu}@sjtu.edu.cn

[2]Microsoft Research Asia
5F, Sigma Center, 49 Zhichun Road
Beijing 100080, P. R. China
{chenxil, hjzeng, zhengc}@microsoft.com

ABSTRACT
The emergence of personalized homepage services, e.g. personalized Google Homepage and Microsoft Windows Live, has enabled Web users to select Web contents of interest and to aggregate them in a single Web page. The web contents are often predefined content blocks provided by the service providers. However, it involves intensive manual efforts to define the content blocks and maintain the information in it. In this paper, we propose a novel personalized homepage system, called "Homepage Live", to allow end users to use drag-and-drop actions to collect their favorite Web content blocks from existing Web pages and organize them in a single page. Moreover, Homepage Live automatically traces the changes of blocks with the evolvement of the container pages by measuring the tree edit distance of the selected blocks. By exploiting the immutable elements of Web pages, the tracing algorithm performance is significantly improved. The experimental results demonstrate the effectiveness and efficiency of our algorithm.

Categories and Subject Descriptors
H.1.2 [**Models and Principles**]: User/Machine Systems – *Human factors, Human information processing*; H.5.4 [**Information Interfaces and Presentation**]: Hypertext/Hypermedia – *Navigation*, *User issues*

General Terms
Algorithms, Experimentation, Human Factors

Keywords
Adaptive User Interfaces, Tracing, Web Blocks, Tree Edit Distance, Tree Pruning.

1. INTRODUCTION
Web users often want to save shortcuts to their interesting information on the Web for convenient re-use. One conventional way is to use Favorites folders to organize the URLs of interest so that they could be visited quickly next time. Some tools are also developed to help users organize their shortcuts, such as Mind-It[1]. However, users may only be interested in some parts of the pages instead of the whole pages. In order to fulfill this requirement, various personalized homepage applications emerged recently on the Web to enable Web users to select Web contents of interest and to customize layouts and visual styles. For example, personalized Google Homepage [2] allows users with Google accounts to consolidate various Google features, ranging from stocks, weather, quote, to search and email, into a personalized homepage. Microsoft Windows Live[3], enables passport users to organize their homepage by collecting their favorite information, including Microsoft services, gadgets, or any RSS feeds. Figure 1 illustrates a screenshot of Windows Live. Similar ideas [2] are also proposed by some researchers using the name of "one-stop browsing".

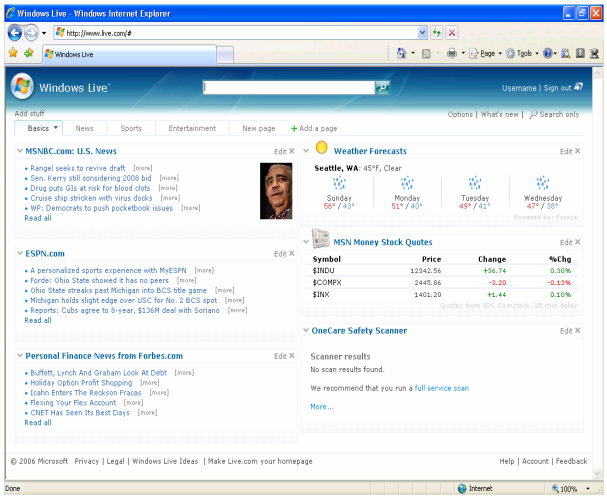

Figure 1. A Screenshot of Microsoft Windows Live Service

Unfortunately, it is not trivial to define the content blocks and to maintain the information in it. Service providers often define their own markup specification and organize their contents by manual effort. Information such as news and blogs often update rapidly, which make the maintenance of these content blocks difficult and expensive.

* This work is conducted during the first author's internship in Microsoft Research Asia.

WWW 2007, May 8–12, 2007, Banff, Alberta, Canada.
ACM 978-1-59593-654-7/07/0005.

[1] NetMind. http://www.netmind.com/.

[2] http://www.google.com/ig.

[3] http://www.live.com/.

In this paper, we propose "Homepage Live", which allow end users to easily collect their favorite Web content blocks from existing Web pages and organize them in a single page. Some previous work [5] [7] [9][17] has shown that a Web page can be partitioned into multiple semantically coherent blocks. Homepage Live automatically recognize these blocks and allows end users to use drag-and-drop actions to select the ones of interest.

A more challenging problem for Homepage Live is how to trace a content block because many Web pages keep updating every day, especially the ones of newsletter. For example, Google News[4] updates the news portal averagely once half an hour. Contents may be inserted and removed; the layouts of Web pages also change frequently. We leverage the Web page DOM tree structure to trace the block. By parsing a Web page into a DOM tree, each block to be traced is represented as a sub-tree. Therefore the block tracing problem is defined as to identify a certain sub-tree in an updated DOM tree of a new page. The intuitive solution is to identify the block by comparing the sub-trees one by one through the general tree edit distance algorithm. The time complexity of the algorithm is $O(N^2D^2)$ where N is the node number of the whole tree and D is the maximum child number of nodes in the whole tree. It is quite time-consuming to trace several blocks simultaneously, especially when some huge pages are involved. Since our problem is to trace the block in two sequential Web pages, there may exist some nodes with unchanged attributes to provide the hints for tracing. We propose an enhanced edit distance algorithm by utilizing such information. First, all nodes with their attributes in two trees are indexed and a fast matching algorithms is used to find the common nodes in two trees. Then, two trees are pruned into reduced trees. Finally, we apply tree edit distance algorithm on the reduced trees. The time cost for tree edit distance algorithm in our proposed algorithm is reduced to $O(N'^2D'^2)$, while N' and D' is much smaller than original ones.

Based on the proposed tracing algorithm, we built Homepage Live to enable Web users to mark blocks from different Web pages and to organize the layout of these blocks. When the pages are updated, the application can automatically trace the marked blocks and show the new version to the users.

The rest of this paper is organized as follows: In Section 2 we discuss the related work. In Section 3 we demonstrate the demo system, and formulate the tracing problem in Section 4. We then introduce two simple algorithms of low accuracy and our advanced algorithms in Section 5 and 6 respectively. In Section 7 we present the experimental evaluation. A case study is given in Section 8. Finally, we conclude the paper in Section 9.

2. RELATED WORK

There are many Web monitoring tools designed and developed since the early 1990s. For example, Mind-It[5] is one of such tools. Users can register URLs of their interests and get notified by email when the pages are updated. [9] proposes Web Tracker which uses the Unix *diff* tools to show the difference to users. Similar to our work, the Do-I-Care agent [1] employs relevance feedback to detect the users' interests. [6] and [9] adapts Hirschberg's solution for finding the longest common subsequence to HTML pages. ChangeDetector™ [3] is a site level Web monitoring tool that can potentially be used to discover "silent news" hidden under corporate Web sites. All these work emphasize on the evolution of Web pages at the granularity of page level or site level. Different from them, our algorithm adopts DOM tree mapping methods at block level.

A Web page can be divided into a set of blocks with different kind of information. Currently, research on Web blocks has become more active, which has many potential applications such as block based Web search [4]. A variety of approaches have been suggested for segmenting a Web page into blocks. These approaches, e.g. DOM-based segmentation [7], location-based segmentation [9], and vision-based page segmentation [5][18], are distinguished from each other by taking various factors as partition criteria. Though these methods have considered the structure of a Web page instead of treating it as a whole unit, they only segment statically. They do not discuss the change of these blocks during the evolution of the Web page, which happens frequently on the Web. In our work, we dynamically trace the block in a Web page after a user marks a block. [16] proposes a hierarchical and fragment-aware model of dynamic Web pages and considers the lifetime of fragments. It aims at detecting fragments that are most beneficial to caching and content generation while our work aims towards tracing the block interested by users.

The idea of building a "one-stop browsing" application is also adopted in Internet scrapbook [18], Stuff I've Seen [9], Web Montage [2] and WebViews [12]. Internet scrapbook is a system which tackles the similar problem to ours. The system enables users to collect content blocks on Web pages and trace it in updated versions. It exploits the plan html tags and contents over the Web pages to trace the target block. Stuff I've Seen [9] provides a unified index of information that a person read, such as emails, Web pages, documents, etc. By a query interface, the system can efficiently find all the related information that the user has ever seen. It regards Web pages as a unit for searching, which is different from the idea of Web Montage's and ours. Web Montage [2] is quite similar to our application. The difference is that they do not exploit the DOM tree of HTML pages while we do. They trace the Web page blocks by recording the size and the position on the distal page of the original block. The method is just too simple for dynamically changing Web pages. If the block size or position is changed, the users have to modify the records. Similar to them, WebViews [12] uses a simple method based on recording the node path of the HTML DOM tree, named XPath to trace the blocks.

Some applications aiming at data records extraction also adopt DOM tree analyzing approaches. Hasan et al. have encoded the DOM tree path by a method similar to regular expressions [8]. Bing et al. have proposed to use tree edit distance to find the similar data records on one Web page [15][21].. Since their targets are different from ours, their methods can not be applied directly to solve our problem. Although the method we proposed in Section 6 is also based on tree edit distance, it is still different from what Bing et al. proposed and we have optimized it.

3. HOMEPAGE LIVE

Homepage Live is an application which offers "one-stop browsing" for users. It allows users to collect blocks from different Web pages and organize them in a single sheet as a personalized page. Figure 2 illustrates a snapshot of Homepage Live. The sheet in the right panel is composed of blocks from different sites.

[4] Google News. http://news.google.com/.

[5] http://www.netmind.com/

Figure 2. Homepage Live

In contrast to identifying a Web page with its URL, our system needs to support the users to mark a Web page block. Our system can help users outline the blocks they want from Web pages, and trace them. When a user re-opens the system, it will automatically access all these Web pages, detect the blocks' positions in the pages by an efficient tracing algorithm, and present the extracted real time content to the user.

To sum up, running a personalized homepage is a two-step process in Homepage Live. First, the users collect the blocks from different Web pages to construct their personalized homepage. Second, the system display the collected Web blocks and automatically update the contents according to the changes of corresponding Web pages.

3.1 Collecting the Blocks

Different from the Web montage [2], we develop a manual block marking tool in Homepage Live to help users collect the blocks. When browsing, the Web page is first parsed into a DOM tree. Each content block in the Web page is mapped to a node in the DOM tree. Then, the tool enables the user to select the block through the mouse operating in different ways:

1) The user can move the mouse on the Web page to select a block and the selected block will be marked by a red rectangle.

2) The user can scroll the wheel of the mouse to change the granularity of the selected block, which is shown in Figure 3. An up-scroll means the parent node is the target block and a down-scroll means the child node under the current cursor is the target block.

3) After the block is confirmed by double-clicking the mouse, the user can drag the block into the personalized page in Homepage Live.

4) The user can organize the layout and the visual style of personalized pages by drag-and-drop.

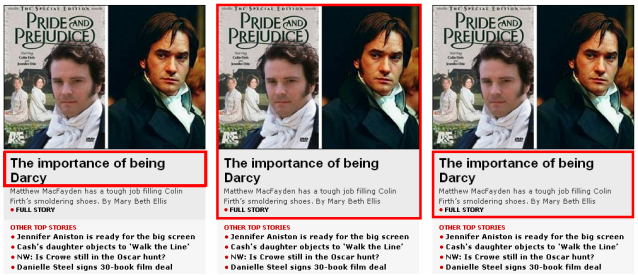

Figure 3. Block Selection

3.2 Tracing Web Page Blocks

When the user runs the application again, the application uses a tracing algorithm to analyze the original pages and the new pages. It can detect the new block position in the updated pages, and present the extracted new blocks to the user. Take Figure 4 as an example, if the user is only interested in the NYSE Composite chart block of the page, the application will fetch the page and show the latest charts.

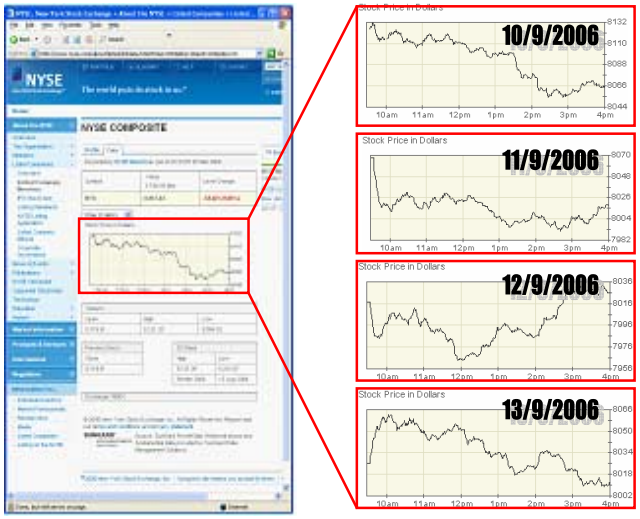

Figure 4. An Example of Sequential Blocks

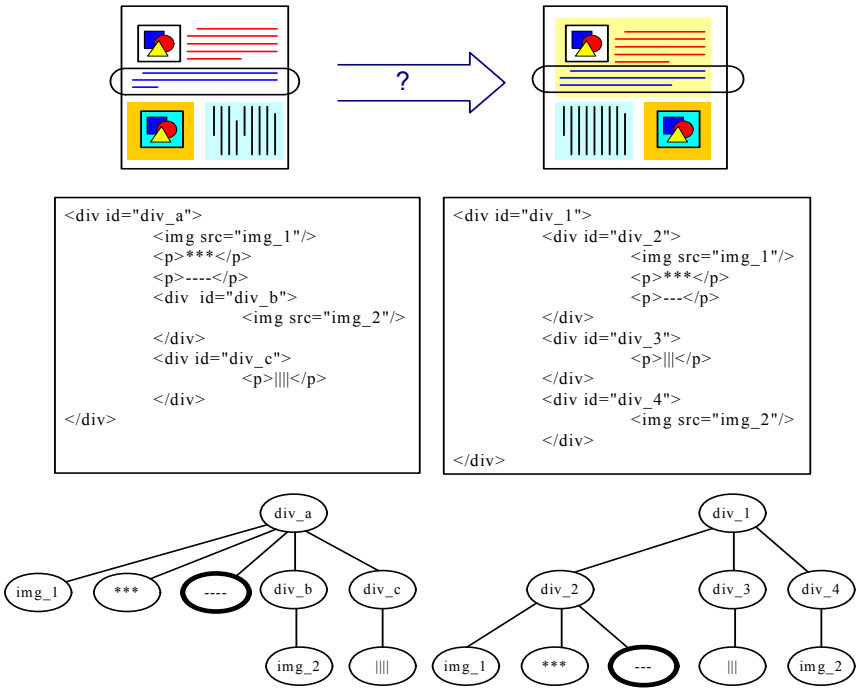

Figure 5. DOM Tree of Web pages

In this paper, we emphasize on the problem of how to trace the block when the content of the corresponding block is changed. Since Web pages are often updated to attract users, their content and layout are also frequently changing. In this case, the tracing problem is a challenging job as we may not just record the visual position and size of the block.

In the following sections, we will define the problem and focus on solving it.

4. TRACING PROBLEM DEFINITION

4.1 Definition

Given a Web page and a user's target block, our problem is to find the corresponding block in the evolved page. The term "corresponding" is to describe the relationship between the users' target block in the evolved page and its original version. The layouts and contents are represented by DOM (Document Object Model) tree. Each HTML page corresponds to a DOM tree where tags are internal nodes and the detailed texts, images or hyperlinks are the leaf nodes. Figure 5 shows two pages with two segments of their HTML codes and their corresponding DOM trees. The right tree is an updated version of the left one.

Formally, given a DOM tree T_o and its sub-tree B_o, when T_o has evolved to T_e, the problem is to find out the evolved block B_e in T_e, corresponding to B_o. B_e should be unique if it exists.

4.2 Difficulty of the Problem

From first glance, the tracing is an easy problem. In fact, due to the diversity of Web pages, the original page could be edited in various kinds of styles. We had a study on a 25-URL data set. The URLs are frequently updating pages, such as Google news, Yahoo news, MSNBC news, etc. For each URL we crawl 101 versions of these URLs with an interval of thirty minutes.

Table 1. Statistics for Web Page Evolution

Item	Average	Distribution	
Node Number	1171	< 1000	52%
		1000 - 2000	40%
		> 2000	8%
Structure Variation after an Hour	11.4%	> 30%	16%
		%0 - 30%	12%
		5% - 10%	44%
		< 5%	28%
Content Variation after an Hour	81.7%	70% - 80%	36%
		80% - 90%	64%
Accumulated Structure Variation	34.0%	> 90%	16%
		30% - 90%	20%
		15% - 30%	24%
		5% - 15%	20%
		< 5%	20%
Accumulated Content Variation	88.5%	> 90%	32%
		85% - 90%	52%
		75% - 85%	16%

Table 1 lists the measurement of our study. The node number of a Web Page ranges from several hundreds to several thousands and averaged 1171. Averagely speaking, the structure and contents vary 11.4% and 81.7% hourly respectively and finally the variation accumulate to 34% and 88.5% respectively. Here structure variation stands for the percentage of deleted, new added and tag-updated nodes over all the nodes, while the contents variation stands for the percentage of content-updated nodes over all the nodes. The data means that a large portion of page structure and most of the content will be changed during the Web Page evolution. Besides, from the distribution analysis, we can observe the diversity of Web Pages. Some of the pages change a lot and some of them are just slightly adjusted in structure.

To sum up, the difficulty of the problem comes from the diversity of Web Pages and the tremendous changes during the Web Page evolution.

5. SIMPLE METHODS

There are some simple methods to solve the problem. Here we introduce two methods, direct path finding and tag string matching.

5.1 Direct Path Finding

Direct Path Finding just records the tags on the path from the root node of T_o to the root node of B_o. When tracing the evolved block, it goes through the recorded path. If there is any different tag on the path, it fails to find the target block in the evolved page. Otherwise, it returns the block in the evolved page which shares the same path with the original one.

This simple method can not deal with the problem of block position changing. If a block moves from the left to the right column, the method will fail to find it correctly.

5.2 Tag String Matching

Another intuitive method is string matching. Tag string matching method encodes B_o by the tag sequence of its preorder traversal. To find the evolved block, it compares the original tag sequence with the tag sequence of every sub-tree of T_e. The similarity of the two tag sequences is the length of the longest common sub-sequences (LCS). The sub-tree of T_e that has the largest LCS value is recognized as B_e. The problem of this algorithm is that there may exist several blocks that are similar to the original one.

Besides , flattening the tree structure into sequence will lose useful information and decrease the precision a lot.

6. TREE MAPPING ALGORITHMS FOR BLOCK TRACING

6.1 Tree Mapping for Block Tracing

String edit distance is not suitable for this problem as it does not consider the tree structure. After string matching, it is hard to decide which matching is the correct one as there are many possible matching. In this paper, we exploit the tree structure and propose to use labeled tree mapping to solve the tracing problem. Through the mapping between the DOM trees of pages, we can get the corresponding nodes between two trees,

Definition 1 A Labeled Tree is a tree with a label l attached to each of its nodes.

DOM tree of Web pages can be transformed into labeled tree by regarding tags as labels.

Definition 2 Let $T[i]$ be the i^{th} node of labeled tree T in a preorder walk. Let $l[r]$ be the label of node r. A mapping M between two labeled trees T and T' is a set of pairs (i, j), one from each tree, satisfying the following conditions for all (i_1, j_1), $(i_2, j_2) \in M$:

(1) $i_1 = i_2$ iff $j_1 = j_2$;

(2) $T[i_1]$ is an ancestor of $T[i_2]$ iff $T'[j_1]$ is an ancestor of $T'[j_2]$.

(3) $l[T[i_1]] = l[T'[j_1]]$, $l[T[i_2]] = l[T'[j_2]]$

Intuitively, the definition requires each node to appear no more than once in a mapping. The hierarchical relation among the nodes is also preserved.

There exist many mappings between two labeled trees. To evaluate the mapping quality, we use *edit distance*.

Definition 3 The *edit distance* between two trees T and T' is the number of unmapped nodes in the two trees.

Our definition of edit distance differs from the one noted in [19] to fit this problem better. Edit distance reflects the cost associated with the minimal set of operations required to transform T into T'. By finding a mapping with minimum edit distance, we can attain the evolved block.

However, find such a mapping between labeled trees is an NP-Complete Problem, as mentioned in [22]. Fortunately, the mapping finding problem can be solved efficiently by the restriction on the order of nodes. Furthermore, as a fact, elements of Web page DOM trees do have order.

Definition 4 An Ordered Tree is a tree with children of each node ordered.

Definition 5 Let $T[i]$ be the i^{th} node of ordered labeled tree T in a preorder walk. Let $l[r]$ be the label of node r. A mapping M between an ordered labeled tree T of size n_1 and an ordered labeled tree T' of size n_2 is a set of pairs (i, j), one from each tree, satisfying the following conditions for all (i_1, j_1), $(i_2, j_2) \in M$:

(1) $i_1 = i_2$ iff $j_1 = j_2$;

(2) $T[i_1]$ is an ancestor of $T[i_2]$ iff $T'[j_1]$ is an ancestor of $T'[j_2]$.

(3) $l[T[i_1]] = l[T'[j_1]]$, $l[T[i_2]] = l[T'[j_2]]$

(4) $T[i_1]$ is on the left of $T[i_2]$ iff $T'[j_1]$ is on the left $T'[j_2]$;

The order between sibling nodes is preserved in the mapping of ordered labeled tree as a supplement of the preservation of hierarchical relation .

Now we propose the basic tree mapping algorithm to minimize edit distance between the new DOM tree and the original one in order to find target node in the new tree. Let t and t' are the root nodes of tree T and T' respectively, n(T) stands for the number of nodes in T. The edit distance is accumulated by the number of unmapped nodes recursively in the two trees as follows:

1. All nodes in T are not mapped to a node in T', then

$$Dis(T, T') = n(T) + n(T')$$

   Intuitively, the edit distance of unmapped sub-trees is the total number of their nodes.

2. If r is mapped to r', the edit distance is the total number of the two trees minus the matched nodes. We assume that p_i and p_i' are monotonically increasing, so that standard dynamic programming algorithm can be used to calculate the mapping with minimum edit distance. Assume that there are m pairs $(S_{pi}, S_{pi'})$ of sub-trees mapped as shown in Figure 6, then

$$Dis(T, T') = n(T) + n(T') - 2 - \sum_{0 \le i < m} (n(S_{pi}) + n(S'_{pi'}) - Dis(S_{pi}, S'_{pi'}))$$

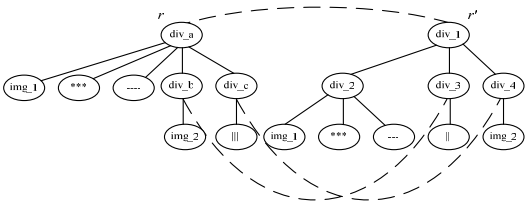

Figure 6. An Example of Tree Matching by Condition 2

3. If r is mapped to the root node s' of sub-tree S' in T', the edit distance of the two trees is the edit distance of T and S' in addition with the unmatched nodes in T'. As shown in Figure 7,

$$Dis(T,T') = n(T') - n(S') + Dis(T,S')$$

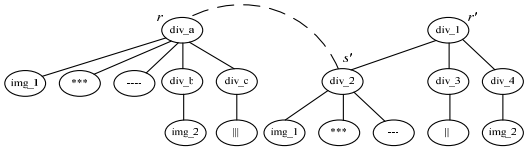

Figure 7. An Example of Tree Matching by Condition 3

Now the edit distance $Dis(T,T')$ is well defined recursively. This algorithm, listed in Table 2, calculates the minimum edit distance. The counterpart mapping can be worked out easily by a similar process.

However, the shortcoming of the algorithm is that the high computational cost. The time complexity of this algorithm is $O(N^2D^2)$, where N is $\max(n(T),n(T'))$ and D is maximum children number of tree nodes. It could keep users waiting for minutes to get the mapping results for large Web pages..

Table 2. General Minimum Edit Distance Mapping

```
int Dis(T, T') {
  r ← root of T;
  r' ← root of T'
  S1,S2…Sx ← the sub-trees of T,
  S'1,S'2…S'y ← the sub-trees of T'
  //case 1
  minDis ← n(T) + n(T');
  //case 2
  if (r.tag() ==  r'.tag()) {
    childrenDis[0][0] ← 0;
    for (i=0; i<=x; i++)
      for (j=0; j<=y; j++)
        if (i>0 || j>0) {
          childrenDis[i][j] ← min (
            Dis(Si, S'j) + childrenDis[i-1][j-1],
            childrenDis[i][j-1],
            childrenDis[i-1][j]
          );
        }
        if (childrenDis[m][n] < minDis)
          minDis ← childrenDis[m][n];
  }
  //case 3
  for (i=0; i<=x; i++) {
    if (Dis(Si, T') + (n(T) - n(Si)) < minDis)
      minDis ← Dis(Si, T') + (n(T) - n(Si));
  }
  for (j=0; j<=y; j++) {
    if (Dis(T, Sj') + (n(T') - n(S'j)) < minDis)
      minDis ← Dis(T, Sj') + (n(T') - n(S'j));
  }
  return minDis;
}
```

6.2 Fixed Sub-tree Based Tracing

Although in Web pages, layout information is more immutable than page content, there are still some tag attributes unchanged during the evolution, including captions of small blocks, some hyperlinks and some images. With the help of these immutable elements, some sub-trees can be pruned away. After that, by performing the algorithm on the reduced tree, the total time cost could be greatly reduced. For this purpose, we introduce an improved algorithm based on tree mapping in this section.

Here we define Fixed Node and Common sub-tree for the sake of convenience in describing our algorithm.

Definition 6 A Fix Node is a node with both tag and attributes immutable in two trees. In our real implementation, we have an extra restriction that the content word length of Fix Node will be no more than 2.

Definition 7 A Common Sub-Tree Pair is a sub-tree pair which satisfies:

1) the sub-tree roots are the same Fix Nodes or

2) the two sub-trees contain a same set of Fix Nodes; and none of their sub-trees contain all the Fix Nodes.

For example, in Figure 7, div_b and div_4 is a common sub-tree pair but div_a and div_2 is not since the fix node img_2 is not includes in div_2.

A Minimum Common Sub-Tree is the common sub-tree with minimum size.

6.3 Algorithm Flow

In this section we will describe *Fixed Sub-Tree Based Tracing* algorithm in detail. It is divided into three steps: finding Fix Nodes; pruning away Fix Nodes and generating reduced trees; and performing Minimum Edit Distance Mapping algorithms on the reduced tree. The algorithm aims to reduce the uncertainty of tree mapping by eliminating some definitely mapped nodes. Thus the time complexity is reduced and the precision is improved.

6.3.1 Finding Fix Nodes

First, all the tags and contents of the nodes in the original tree are indexed, for example by a binary tree or a hash table. Duplicated nodes, with the same tags and contents, are removed for disambiguation purpose. Then, for the evolved tree, we check all nodes sequentially and pick out the nodes whose content appears in the original tree.

Table 3. Finding Minimum Common Tree

```
TreePair findMinCommonTree() {
  S ← the traced block;
  Fn ← the set of Fix Nodes;
  while (S != T && S ∉ Fn) {
    //two Fix Nodes' lowest common ancestor
    if (S has more than two sub-trees
      which have Fix Node)
      break;
    S ← S's direct super tree;
  }
  if (S = T) return ( <T,T'> );
  if (S ∈ Fn){
    S' ← corresponding Fix Node in T'
    return ( <S,S'> );
  }
  Fn(S) ← the set of Fix Nodes in S;
  S' ← (T')'s the lowest sub-tree
          which contains Fn(S);
  Fn(S') ← the set of Fix Nodes in S';
  While (Fn(S) != Fn(S') ){
    if (Fn(S') ⊇ Fn(S) ){
      S ← S's direct super tree
      Fn(S) ← the set of Fix Nodes in S;
    }
    else {
      S' ← The direct super tree of S'
      Fn(S') ← the set of Fix Nodes in S';
    }
  }
  return ( <S,S'> );
}
```

6.3.2 *Generating the Reduced Trees*

In this algorithm, only a subset of the nodes in the minimum common sub-tree which contains the tracing block will be taken into consideration in the mapping phase. We call it the reduced tree. The ancestors and sibling-trees of the minimum common sub-tree will be cut off, as mapping between them is unnecessary.

The reduced tree is built by the following process: First, our algorithm finds the minimum common tree pair contains the tracing blocks with the following algorithm in Table 3. After the minimum common tree pair is found, the algorithm then prunes away some sub-trees that are intuitively unnecessary to be taken into consideration in the matching phase, in a rule based fashion. For each Fix Node, all of its ancestor nodes, except the nodes lies in the path from the root to the tracing block, should be cut off. However, in the pruning process, all the nodes from the root to the tracing block as well as the successors should be preserved, because they are always useful in the matching phase.

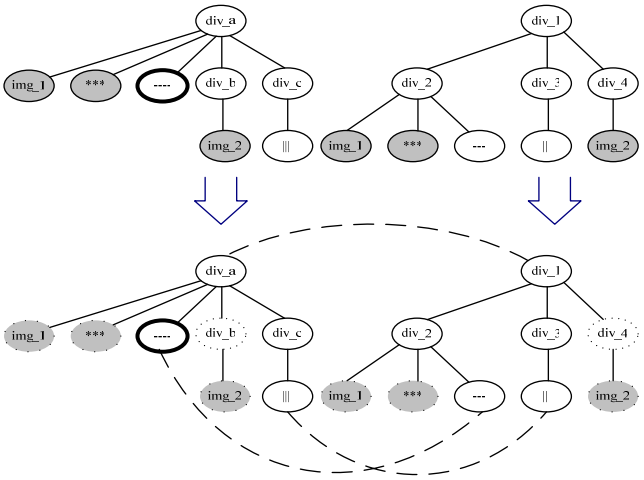

Figure 8. An illustration of fixed sub-tree based tracing mapping

6.3.3 *Mapping on the Reduced Trees*

Since the scale of DOM trees has been greatly reduced, only the remaining nodes in the minimum common sub-tree will be taken into consideration in this phase by minimum edit distance algorithm.

Figure 8 shows an illustration of Fixed Sub-tree Based Tracing. We mark the block with a bold circle in the left tree and want to trace the corresponding nodes in the right tree. In the first step, we pick out Fix Nodes, which are marked with shadows. In the second step, we find that the minimum common sub-tree pair. In this case, it is the pair of two whole trees. Then, we cut off all the path contains Fix Nodes. The pruned nodes are embraced by broken line in the bottom part of Figure 8. As a result, there will be only four nodes in each reduced tree. After that, we perform minimum edit distance algorithm and finally find the target block.

7. EXPERIMENTS AND ANALYSIS

This section provides empirical evidence about the accuracy and usability of Homepage Live. The experimental results show the effectiveness of our algorithm.

7.1 Data Set and Metrics

An experiment has been performed on a 25-url dataset to test the effectiveness of our algorithm. These URLs are pages of Google news, Yahoo news, MSNBC news, etc. All of them are frequently updated, once about twenty minutes on the average. By recording a version every thirty minutes, we get 101 pages for each URL. Five users are asked to select their interested blocks of the first version of 25 URLs, one block for each page. They are asked to mark out the evolved blocks in the later 100 versions also. The evolved blocks can be nothing since sometimes there are no proper corresponding blocks. So, there are totally 125 block cases traced. Each block has been traced 100 times. In total, there are 12,625 blocks marked.

Two metrics have been used to measure the effectiveness of tracing the blocks. One is Correct Tracing Rate (CTR); the other is Correct Case Rate (CCR). CTR is the percentage of the correct tracing count of the total tracing count. In our experiments, there are 12,500 traces. CCR is the percentage of the correct case count of the total case count. In our experiments, each block is regarded as a case and we only regard the whole case as correct if all the 100 tracings of the block are successful. The total number of cases in our experiment is 125.

Table 4. CTR and CCR of the Four Methods

Method	DPF	TSM	TED	FSBP
CTR	0.95	0.94	0.98	0.98
CCR	0.75	0.71	0.87	0.87

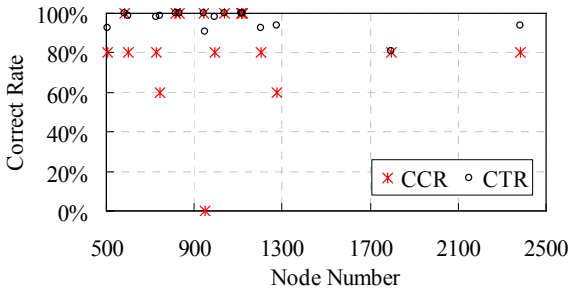

Figure 9. Correct Rate of FSBP vs. HTML DOM Tree Node Number.

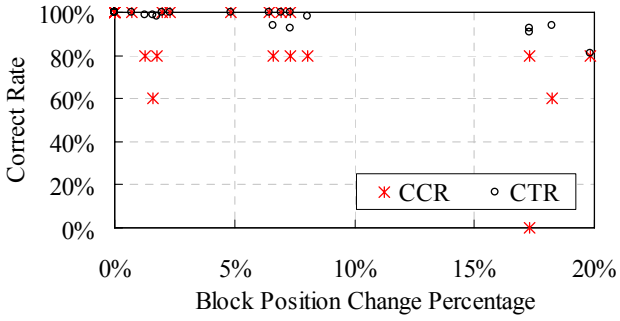

Figure 10. Correct Rate of FSBP vs. Block Position Change Percentage

7.2 Effectiveness

We have applied four methods on the data set. They are: Direct Path Finding (DPF), Tree String Matching (TSM), Tree Edit Distance (TED) and Fix Sub-tree Based Tracing (FSBT). Each method is implemented as their definition in Section 5, Section 6 and Section 7 respectively. The CTR and CCR values are listed in Table 4. It is clear that TED and FSBP have the best effectiveness. They do not beat each other in the data set. That is to say, compared with TED, FSBT did not lose precision in our experiment while it improved computational performance, which will be listed in the next sub-section.

Figure 9 shows the relation between the correct rate of FSBP and the HTML DOM tree node number. Figure 10 shows the relation between the correct rate of FSBP and the block position change percentage. The data is collected based on the 25 URLs, i.e. for each URL, we calculated their corresponding CCR and CTR. Since there are 5 cases on each page, the value of CCR is 0%, 20%, 40%, 60%, 80% or 100% and the CTR varies from 0% to 100% with the interval of 1/500. The block position change percentage also varies from 0% to 100% with the step of 1/500, which is calculated as the division on the number of position changed blocks in the 100 evolved pages by the total number 500.

From the charts, we do not find a clear drop of correct rates when the page DOM tree node number grows or the block position change percentage grows. That is to say, the size and the change rate of the Web page do not impact our algorithm much. Thus, these two figures proved the scalability of our algorithm.

7.3 Computational Cost

The computational cost can be measured by the time and memory space needed for block tracing. We have compared the costs of TED and FSBP.

The experiment result is partially listed in table 5. In table 5 we list the average time cost and memory cost of TED and FSBP. The time is measured by millisecond and the metrics of memory is kilo-byte. From the table it's clear that FSBP outperforms TED coherently, in terms of both time and memory. Averagely speaking, FSBP saves around 18% of time and 55% of memory.

Table 5. Computational Cost of TED and FSBP

TED Time	FSBP Time	Ted Memory	FSBP Memory
921	755	57231	25583

Intuitively, the computational cost of TED is in proportion to the tree size, while that of FSBP approximately depends on the size of the reduced tree. Table 6 lists some of our experiment data. The columns means the tree size, marked block size, the number of fix nodes, the size of minimum common tree and the size of reduced tree respectively. From the table we can find that the size of reduced tree relies much on the size of tracing block but not the size of the whole tree. Therefore, the increasing size of trees does not impact the computational cost of FSBP significantly.

Table 6. Relation of size between tree and reduced tree

Tree	Block Size	Fix Node	Minimun Common Tree	Reduced Tree
1637	1	99	179	5
822	1	86	17	12
1222	4	41	198	32
826	15	49	20	16
800	17	83	18	18
1222	25	41	201	59
800	30	49	172	70
1222	35	41	139	50
802	42	74	82	70
1647	54	110	182	59

8. CASE STUDY

As a case study, we compare the two versions of MSN news (http://www.msnbc.com) to show the performance of the algorithm. The latter version is updated 24 hours later than the former.

The first case indicates the adaptation of our algorithm when the content of the block is changed. As shown in Figure 11, an extra item is added to the original page and the titles of the rest items are also changed. Our algorithm can trace the changed block according to the structure of the two Web pages since the root of traced block is a Fix Node.

In the second case, we show the case when the order of the sub-nodes in a block is changed and our algorithm can also work well. As shown in Figure 11, the traced block is consisted of two parts: image and description parts. The description is put in right side of block in the original page while it is changed to the left side in the evolved page. Since our algorithm considers that the inner structure of the two sub-trees is little changed and the position of block in the whole page tree structure is little changed, it can still trace such block through the whole tree matching algorithm.

In these two cases, direct path finding may work properly and the tree string matching method does not work. Moreover, they both fail in case 3 because the confusion of two blocks in the original page, "more on decision 2006" and "more top stories". The same path and sub-tree structure of them infer the confusion. However, with the help of the label "more top stories", we identified it as a Fix Node and therefore find the target block.

9. CONCLUSION

In this paper, we have introduced a novel application, Homepage Live, for tracing interesting blocks on different Web pages in a single display. The application adopts a novel method where tree edit distance is utilized to trace the block when the page is updated. By exploiting the immutable elements of Web pages, we decrease the time complexity and space complexity a lot. When given two Web pages, the method first recognizes the immutable nodes in each DOM tree. Then, the DOM trees are pruned into a reduced tree with removing the unchanged nodes. Finally, the fast tree matching algorithm is applied to trace the target block. The experimental results show that the tracing precision is much higher than direct path finding and tree string matching. It has achieved a 98% correct tracing rate and an 87% correct case rate.

With the ability of automatic recognizing and tracing Web blocks, we are able to develop some sections or gadgets for personalized homepage applications, e.g. Google Homepage and Microsoft Windows Live, as an additional choice of the limited candidate pieces of information developed by the service providers themselves.

10. REFERENCES

[1] Ackerman, M., Starr, B. and Pazzani, M., The Do-I-Care Agent: Effective Social Discovery and Filtering on the Web. In *Proceedings of RIAO'97*, 17-31.

[2] Anderson, C. R. and Horvitz, E.. Web montage: a dynamic personalized start page. In *Proceedings of the Eleventh International Conference on World Wide Web*, pages 704--712. ACM Press, 2002.

[3] Boyapati, V., Chevrier, K., Finkel, A., Glance, N., Pierce, T., Stockton, R. and Whitmer, C.. ChangeDetector™: A Site-Level Monitoring Tool for the WWW. In *Proceedins of 11th International World Wide Web Conference (WWW 2002)*, 2002, 570-579.

[4] Cai, D., Yu, S.P., Wen, J.R. and Ma, W.Y.. Block-based Web search. In *Proceedings of the 27th annual International Conference on Research and Development in Information Retrieval (SIGIR 2004)*, 2004, ACM Press, 456-463.

[5] Cai, D., Yu, S.P., Wen, J.R. and Ma, W.Y.. VIPS: a vision-based page segmentation algorithm. Microsoft Technical Report, MSR-TR-2003-79, 2003.

[6] Chen, Y.F., Douglis, F., Huan, H. and Vo, K.P., TopBlend: An Efficient Implementation of HtmlDiff in Java. In *Proceedings of the WebNet 2000 Conference*, San Antonio, TX, Nov. 2000.

[7] Chen, J., Zhou, B., Shi, J., Zhang, H.J. and Qiu, F.. Function-Based Object Model Towards Website Adaptation. In *Proceedings of 10th International World Wide Web Conference (WWW 2001)*, 2001, 587-596.

[8] Davulcu, H., Yang, G., Kifer, M., and Ramakrishnan, I. Computational aspects of resilient data extraction from semistructured sources. In *19th ACM Symposium on Principles of Database Systems*, 136--144, 2000.

[9] Douglis, F., Ball, T., Chen, Y., and Koutsofios, E. 1998. The AT&T Internet Difference Engine: Tracking and viewing changes on the web. *World Wide Web* 1, 1 (Jan. 1998), 27-44.

[10] Dumais, S., Cutrell, E., Cadiz, J., Jancke, G., Sarin, R., and Robbins, D. C. 2003. Stuff I've seen: a system for personal information retrieval and re-use. In *Proceedings of the 26th Annual international ACM SIGIR Conference on Research and Development in informaion Retrieval* (Toronto, Canada, July 28 - August 01, 2003). SIGIR '03. ACM Press, New York, NY, 72-79.

[11] Fishkin, K. and Bier, E., WebTracker – a Web Service for tracking documents. *In Proceedings of 6th International World Wide Web Conference (WWW 1997)*, 2004.

[12] Freire, J., Kumar, B., and Lieuwen, D. 2001. WebViews: accessing personalized web content and services. In *Proceedings of the 10th international Conference on World Wide Web* (Hong Kong, Hong Kong, May 01 - 05, 2001). WWW '01. ACM Press, New York, NY, 576-586.

[13] Kovacevic, M., Diligenti, M., Gori, M., and Milutinovic, V. 2002. Recognition of Common Areas in a Web Page Using Visual Information: a possible application in a page classification. In *Proceedings of the 2002 IEEE international Conference on Data Mining (Icdm'02)* (December 09 - 12, 2002). ICDM. IEEE Computer Society, Washington, DC, 250.

[14] Lin, S.H. and Ho, J.M. Discovering Informative Content Blocks from Web Documents. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (SIGKDD 2002)*, 2002.

[15] Liu, B., Grossman, R. and Zhai, Y. Mining Data Records in Web Pages. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD-2003)*, Washington, DC, USA, August 24 - 27, 2003.

[16] Ramaswamy, L., Lyengar, A., Liu, L. and Douglis, F.. *Automatic Detection of Fragments in Dynamically Generated Web Pages. In Proc. of 13th International World Wide Web Conference (WWW 2004)*, 2004, 443-454.

[17] Song, R.H., Liu, H.F., Wen, J.R. and Ma, W.Y.. Learning Block Importance Models for Web Pages. In *Proceedings of 13th International World Wide Web Conference (WWW 2004)*, 2004, 203-211.

[18] Sugiura,A., Koseki,Y. Internet Scrapbook: Automating Web Browsing Tasks by Demonstration. *ACM Symposium on User Interface Software and Technology 1998*: 9-18

[19] Tai. The Tree-to-Tree Correction Problem. *J. ACM 26(3)*: 422-433 (1979)

[20] Yu, S., Cai, D., Wen, J.R. and Ma, W.Y.. Improving Pseudo-Relevance Feedback in Web Information Retrieval Using Web Page Segmentation. In *Proceedings of 12th International World Wide Web Conference (WWW 2003)*, 2003, 11-18.

[21] Zhai, Y., and Liu, B.. Web Data Extraction Based on Partial Tree Alignment, in *Proceedings of the 14th international World Wide Web conference (WWW-2005)*, May 10-14, 2005, in Chiba, Japan.

[22] Zhang, K., Statman, R. and Shasha, D. On the editing distance between unordered labeled trees. *Information Processing Letters*, 42(3):133–139, 1992.

Figure 11. Case Study