Automatically extracting user reviews from forum sites

Wei Liu a, Hualiang Yan b, Jianguo Xiao b,∗

a Institute of Scientific and Technical Information of China, Peking, 100038, China
b Institute of Computer Science & Technology, Peking University, 100871, China

ARTICLE INFO

Keywords:
Web reviews
Structured data
Web data extraction

ABSTRACT

User reviews in forum sites are the important information source for many popular applications (e.g., monitoring and analysis of public opinion), which are usually represented in form of structured records. To the best of our knowledge, little existing work reported in the literature has systematically investigated the problem of extracting user reviews from forum sites. Besides the variety of web page templates, user-generated reviews raise two new challenges. First, the inconsistency of review contents in terms of both the document object model (DOM) tree and visual appearance impair the similarity between review records; second, the review content in a review record corresponds to complicated subtrees rather than single nodes in the DOM tree. To tackle these challenges, we present WeRE — a system that performs automatic user review extraction by employing sophisticated techniques. The review records are extracted from web pages based on the proposed level-weighted tree similarity algorithm first, and then the review contents in records are extracted exactly by measuring the node consistency. Our experimental results based on 20 forum sites indicate that WeRE can achieve high extraction accuracy.

© 2011 Elsevier Ltd. All rights reserved.

1. Introduction

With the rapid development of Web 2.0, web users can freely post their reviews for specific events or objects on web pages to voice their opinions. The number of reviews is increasing at a surprising speed, and the reviews cover almost all the domains in the real world, such as commerce, politics, and entertainment. Web reviews are an important information source for many popular web applications, such as monitoring and analysis of public opinion. These applications need an efficient way to harvest large-scale user reviews from many forum sites. Though some efforts [1–3] have been made for this task, most of them focus on crawling review pages. In practice, user reviews need to be extracted from web pages at a fine granularity instead of by web page indexing.

This paper studies the problem of automatically extracting user reviews from forum sites, which consists of two subtasks: (1) review record extraction; (2) review content extraction. Review record extraction detects the boundaries of the review records embedded in web pages and further extracts them as the input of the next subtask. Review content extraction extracts the review contents from the extracted review records. Fig. 1 shows the snapshots of two reviews in one review page from a forum site, and their review contents are also marked with red dashed boxes.

Obviously, it is not practical to generate lots of wrappers manually for different-template review pages. To meet the requirement of real applications, our goal is to propose a template-independent solution which can extract user reviews from different-template review pages without human interaction. It is widely known that web reviews are released by different users rather than the web page owners, so the review contents are usually very inconsistent in terms of both information

∗ Corresponding author.

E-mail address: xjg@ict.pku.edu.cn (J. Xiao).

0898-1221/$ – see front matter © 2011 Elsevier Ltd. All rights reserved.
format and text length. Users are free to use various fonts for emphasis or decoration proposes. Furthermore, images, tables, and video are frequently used in review contents. This distinct character of web reviews makes the extraction task more challenging. Extracting user reviews belongs to the field of web data extraction. Many efforts have contributed to this field. Unfortunately, the existing solutions are not competent for this task due to the inconsistency of the review contents. Compared with other types of web data records (such as the listed records in a search result page), two new challenges are posed by web review extraction. First, the inconsistency of review contents in terms of both the document object model (DOM) tree and visual appearance impairs the similarity between review records. This makes existing solutions, such as the approaches proposed in [4,5], fail to extract review records accurately (see Section 7.3). Second, the review content in a review record is a complicated random subtree rather than a leaf node in the DOM tree, and the tree structures of review contents often vary significantly, even when the review records use the same template. In previous works, the data items to be extracted are leaf nodes in the DOM tree, while review content extraction is performed to detect the minimum subtree which covers the pure review content in the DOM tree.

To tackle these two challenges, we present WeRE — a system that performs automatic extraction of user reviews by employing sophisticated techniques. The review records are extracted from web pages first, and then the review contents in the records are extracted. For review record extraction, a level-weighted tree matching algorithm is proposed to compute the similarity between two subtrees. Based on this algorithm, the noise is removed and the boundaries of the review records are detected. For review content extraction, we measure the consistency of each node of review records in the DOM tree, and further extract the minimum subtree that contains the pure review content based on the node consistencies. In addition, WeRE provides two extraction choices: direct extraction and wrapper-based extraction. For direct extraction, no sample pages are required, and review pages using different templates can be processed together. Its only demand is that each page must contain multiple review records. For wrapper-based extraction, the review record wrapper and the review content wrapper are generated with one sample page by employing direct extraction, and then the wrappers are used to perform review record extraction and review content extraction, respectively. Wrapper-based extraction requires that the review pages are similar in terms of page template and that the sample page must contain multiple review records.

In summary, the contributions in this paper have three aspects. First, we tackle the challenges posed by the inconsistency between review contents. Existing works on web data extraction assume that the items with the same semantics are similar in terms of the DOM tree or visual appearance. Second, we present a lightweight and comprehensive solution for user review extraction. We also provide two extraction choices to meet the requirement of real applications. Third, our scalability and performance experiments show that WeRE can achieve very high extraction accuracy and is very efficient in real applications.

The rest of the paper is organized as follows. The architecture of our solution is presented in Section 2. Section 3 introduces web page representation and extraction features. The methods of review record extraction and review content extraction are proposed in Sections 4 and 5, respectively. Section 6 introduces wrapper-based extraction. The experimental results are reported in Section 7. Related work is reported in Section 8, and Section 9 presents the conclusion.

2. System overview

We have implemented an automatic system, WeRE, for web review extraction. The system architecture of WeRE is shown in Fig. 2. The input of WeRE is a group of review page, and the output is a relational table: each row is the review content of a review record. Users can choose direct extraction or wrapper-based extraction according to the actual inquiries.
The functions of the modules in WeRE are introduced as follows.

**Web page representation:** A review page is parsed into a DOM tree, and the visual information of the tree nodes is attached.

**Review record extraction:** The minimum subtree that contains all review records is detected from the DOM tree first, and then all review records are extracted through removing the noise and detecting the boundaries of the review records.

**Review content extraction:** For review content extraction, the minimum subtree that contains the pure review content of each review record is extracted.

**Direct extraction:** The review pages from different forum sites are inputted indiscriminately, but every page must contain multiple review records.

**Wrapper-based extraction:** The review record wrapper and the review content wrapper are generated by employing direct extraction with one sample page. Then, the review records and the review contents are extracted in turn with their corresponding wrappers.

In the rest of this paper, we will focus on the technical details of these modules.

3. **Web page representation and useful features for extraction**

3.1. **Web page representation**

We parse a review page into a DOM tree, and perform the extraction task in the DOM tree. This approach has been widely adopted by recent works on web data extraction. The related techniques are very mature, and no more discussion of them is given in this paper.

Due to the poor ability of HTML tags with respect to semantic representation, we combine the visual information into the DOM tree to improve the extraction performance. Visual information on web pages has proven to be a very useful feature for web data search and extraction, and several vision-based approaches have been proposed, such as those presented in [4,6,7]. During the parsing process, useful visual information is obtained and attached to the nodes of the DOM tree. The visual information used in this paper is classified into three types, listed as follows.

- **Position:** the coordinate of the left-top corner of a node.
- **Size:** the width and height of the rectangle that a node occupies in the web page.
- **Font:** the fonts of the texts of a node, including font size, font style, and font color.

3.2. **Extraction features**

To ease reader convenience, review pages are always well designed to make the layout of review records regular. Based on our observations of a large number of review pages, a group of useful features is generalized for web review extraction.

To help understand the features, Fig. 3 shows a general case of the review records in web pages and the DOM tree, which is a simplification of a real review page. We use $T_{\text{region}}$ to denote the minimum subtree containing all review records in the DOM tree. Fig. 3(a) is the area of $T_{\text{region}}$ on the web page, and Fig. 3(b) shows $T_{\text{region}}$ in the DOM tree. With the case illustrated in Fig. 3, the useful features are introduced below.

**Visual features**

- **VF1:** Review records are arranged vertically with the same width and flush left.
- **VF2:** The templates of the review records in one page are identical.
- **VF3:** The items with the same semantics in different review records are similar in appearance, including position and font, but the user-generated review is an exception.
Table 1
The validity of the extraction features.

<table>
<thead>
<tr>
<th>Feature Type</th>
<th>VF1</th>
<th>VF2</th>
<th>VF3</th>
<th>VF4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visual features</td>
<td>100%</td>
<td>98.9%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOM tree features</td>
<td>99.1%</td>
<td>99.3%</td>
<td>100%</td>
<td>52.7%</td>
</tr>
</tbody>
</table>

Fig. 3. A general case of the review records in the web page and in the DOM tree.

DOM tree features
- **DTF1**: A review record consists of one or more child subtrees (denoted as $T_{\text{review}}$) of $T_{\text{region}}$. For example, each review record in Fig. 3 consists of two $T_{\text{review}}$s.
- **DTF2**: The review records in one page contain the same number of the $T_{\text{review}}$s. For example, all three review records in Fig. 3 contain two $T_{\text{review}}$s.
- **DTF3**: The semantic order of the subtrees of each review record is consistent, i.e., the $i^{\text{th}}$ subtrees of review records have the same semantics, such as $T_2$, $T_4$, and $T_6$ in Fig. 3.
- **DTF4**: Besides review records, there is often some noise (denoted as $T_{\text{noise}}$) located at the beginning or end of $T_{\text{region}}$, such as “Pages: 1|2 . . . Next”.

To verify the validity of these features, we examine the statistics on a large number of review pages (the data set is introduced in Section 7.1). Table 1 shows the statistics, which are the percentages of review pages that satisfy the features. The statistics indicate that our proposed features are applicable to most of the review pages. There is mainly one case that makes VF2, DTF1, and DTF2 not 100%. This case is review pages that have several top-ranked hot review records, and the template of these hot review records is different to that of other common review records. DTF4 is the percentage of review pages whose $T_{\text{region}}$ contains noise, and this feature indicates that removing the noise is a necessary step during the process of review record extraction.

4. Review record extraction

The process of review record extraction includes three steps: first, $T_{\text{region}}$ is detected in the DOM tree; next, the noise is removed from $T_{\text{region}}$; finally, the boundaries of the review records are detected, i.e., how many sequential subtrees make up one review record.

The process of detecting $T_{\text{region}}$ is very similar to that of detecting search result records group in search result pages studied in [8]. The problem is solved by using four simple visual features of the search result records group: (1) it occupies a large area; (2) it is centrally located; (3) it contains many characters; and (4) it has a large number of records. We adopt this method for detecting $T_{\text{region}}$. Because review records are ordered according to their post dates, we supplement a new feature to improve the accuracy: it contains many ordered dates.

In the rest of this section, we first propose a level-weighted tree similarity algorithm, and then introduce the methods for removing noise and detecting the boundaries of review records based on the proposed algorithm.

4.1. Level-weighted tree similarity algorithm

According to VF2, the review records in one page are presented with one template, so two subtrees with the same semantics should be more similar on the tree structure than those with different semantics. In order to cluster the child subtrees of $T_{\text{region}}$ by their semantics, we propose a level-weighted tree similarity algorithm (LTSA). The basic idea of this
Fig. 4. The overall LTSA.

Algorithm MSM //Max Sequence Matching
Input: \{a1,a2,..,an\}, \{b1,b2,..,bn\}; // two node sequences
Output: s; // a real value
Begin
1. If (m=0) or (n=0) then
   return 0;
2. \(s_0 = MTS(A_1,B_1) + MSM(\{a_2,..,an\},\{b_2,..,bn\})\); // A, and B, are the subtrees whose root nodes are \(a_1\) and \(b_1\) respectively
3. \(s_1 = MSM(\{a_1,a_2,..,an\},\{b_2,..,bn\})\);
4. \(s_2 = MSM(\{a_2,..,an\},\{b_1,b_2,..,bn\})\);
6. return max\{s_0, s_1, s_2\}
End

Algorithm MTS //Max Tree Similarity
Input: A, B; //Two subtrees
Output: s; //The similarity between A and B
Begin
1. a=the root of A;
2. b=the root of B;
3. if NodeMatching(a, b) is false then
   return 0;
5. \(s = NI(a,b) + MSM(Children(a),Children(b))\);
6. return s
End

The detail of the LTSA is shown in Fig. 4. The LTSA consists of two subalgorithms: the max tree similarity (MTS) algorithm and the max sequence matching (MSM) algorithm. We use the MTS algorithm to measure the similarity of any two trees \((A\text{ and } B)\) with complexity \(O(mn)\), where \(m\) and \(n\) are the sizes of the two trees, respectively. The output is their similarity. The MTS algorithm is implemented with a recursive strategy. First, the root nodes of \(A\) and \(B\) are compared (lines 1–6): if they do not match, i.e., their tags are not the same or share no fonts, the process is stopped and the value 0 is assigned as the similarity of the two trees respectively. Next, the similarity of \(A\) and \(B\) is measured; this depends on the level of \(a\) and \(b\) (i.e., \(NI(a,b)\)) and the similarity between their child node sequences (line 6). \(NI(a,b)\) is computed as follows:

\[ NI(a,b) = \exp \left( \frac{\text{level}(a,b) - \text{avgDepth}(T_{\text{region}})}{\text{avgDepth}(T_{\text{region}})} \right). \tag{1} \]

where \(\text{level}(a,b)\) is the path length from the root node of \(T_{\text{region}}\) to \(a\) or \(b\), and \(\text{avgDepth}(T_{\text{region}})\) is the average depth of its subtrees. Obviously, \(NI(a,b)\) is determined by \(\text{level}(a,b)\) and \(\text{avgDepth}(T_{\text{region}})\): the larger \(\text{level}(a,b)\) is or the smaller \(\text{avgDepth}(T_{\text{region}})\) is, the larger \(NI(a,b)\) will be. \(NI(a,b)\) is always a positive number. We do not use \(\text{level}(a,b)\) to denote \(NI(a,b)\) directly by considering the differences of web page templates. The MSM algorithm is used to measure the similarity of two node sequences, which are the children nodes of two trees, by finding their maximum matching through dynamic programming with complexity \(O(mn)\). The node matching metric is the same as that used in the MSM algorithm. Three similarities, \(s_0\), \(s_1\), and \(s_2\), are measured separately (lines 3–5), and the maximum one is returned. \(s_0\) is the sum of the similarity of node pair \((a_1,b_1)\) and the similarity of two sequences \((a_2, \ldots, a_n)\) and \((b_2, \ldots, b_n)\). The former is measured with the MTS algorithm, and the latter is measured with the MSM algorithm, while \(s_1\) and \(s_2\) are the similarity of \((a_1, \ldots, a_m)\) and \((b_2, \ldots, b_n)\) and the similarity of \((a_2, \ldots, a_n)\) and \((b_1, \ldots, b_n)\), respectively.

The similarities of any two child subtrees of \(T_{\text{region}}\) in Fig. 4 being computed with the MTS algorithm are shown in Table 2. From the table, we see that (1) the \(T_{\text{reviews}}\) with same semantics are more similar to each other on the tree structure; (2) the similarity of \(T_{\text{noise}}\) and any other subtree is always a very small number. Motivated by this fact, we propose an algorithm for removing the noise subtrees and an algorithm for detecting the boundaries of review records in turn.

4.2. Removing the noise in the review region

When the similarity of any two child subtrees of \(T_{\text{region}}\) is measured with the MTS algorithm, every child subtree gets \(n-1\) similarities with other \(n-1\) child subtrees. Further, we use \(g_s\) to denote the maximum one of the \(n-1\) similarities.
Table 2
The similarities of any two subtrees of the data region in Fig. 3.

<table>
<thead>
<tr>
<th></th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
<th>T6</th>
<th>T7</th>
<th>T8</th>
<th>GS</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>1.34</td>
<td>1.18</td>
<td>1.02</td>
<td>0.44</td>
<td>1.36</td>
<td>0.75</td>
<td>0.51</td>
<td>1.36</td>
<td></td>
</tr>
<tr>
<td>T2</td>
<td>1.34</td>
<td>1.18</td>
<td>1.02</td>
<td>0.44</td>
<td>1.36</td>
<td>0.75</td>
<td>0.51</td>
<td>1.36</td>
<td></td>
</tr>
<tr>
<td>T3</td>
<td>0.63</td>
<td>0.86</td>
<td>0.86</td>
<td>1.04</td>
<td>0.93</td>
<td>0.98</td>
<td>0.57</td>
<td>6.83</td>
<td></td>
</tr>
<tr>
<td>T4</td>
<td>1.02</td>
<td>6.83</td>
<td>5.49</td>
<td>1.04</td>
<td>0.93</td>
<td>0.98</td>
<td>0.57</td>
<td>6.83</td>
<td></td>
</tr>
<tr>
<td>T5</td>
<td>0.44</td>
<td>1.07</td>
<td>5.49</td>
<td>1.04</td>
<td>0.93</td>
<td>0.98</td>
<td>0.57</td>
<td>6.83</td>
<td></td>
</tr>
<tr>
<td>T6</td>
<td>1.36</td>
<td>7.52</td>
<td>0.92</td>
<td>6.77</td>
<td>1.11</td>
<td>0.62</td>
<td>7.52</td>
<td>8.03</td>
<td></td>
</tr>
<tr>
<td>T7</td>
<td>0.75</td>
<td>0.91</td>
<td>8.03</td>
<td>0.98</td>
<td>5.64</td>
<td>1.11</td>
<td>0.59</td>
<td>8.03</td>
<td></td>
</tr>
<tr>
<td>T8</td>
<td>0.51</td>
<td>0.89</td>
<td>0.66</td>
<td>0.57</td>
<td>0.45</td>
<td>0.62</td>
<td>0.59</td>
<td>0.89</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 5. An example of the change trend of the global similarities in Table 2.

Algorithm RNI // Removing Noise Information
Input: \( T_{\text{region}} \) // The review region
Output: \( R\text{Start}, R\text{End} \) // \( R\text{Start} \) is the start position of the first review record, \( R\text{End} \) is the end position of the last review record
Begin
1. \( T\text{Set} = \text{getChildNodes}(T_{\text{region}}); \)
2. \( \text{tsm}[i][j] = 0; // \text{Initialize matrix TSM, } i = 0, \ldots, n, j = 0, \ldots, n \)
3. for \( i = 1 \) to \( n \) do
4.   for \( j = 1 \) to \( n \) do
5.     if \( i \neq j \) then
6.       \( T[i] = \text{the } r\text{th child sub-tree of } T_{\text{sub}}; \)
7.       \( T[j] = \text{the } r\text{th child sub-tree of } T_{\text{sub}}; \)
8.       \( m[i][j] = \text{MTS}(T[i], T[j]); // \text{the tree similarity of } T[i] \text{ and } T[j] \)
9.     for \( i = 1 \) to \( n \) do
10.    \( gs[i] = \text{Max}\{m[i][j], j = 0, \ldots, n \}; \)
11.    \( R\text{Start} = \text{Max}\{gs[i], i < n\}; // i < n \)
12.    \( R\text{End} = \text{Min}\{gs[i], i < n\}; // i < n \)
13.   If \( R\text{Start} < \lambda \) then /\( \lambda \) is the predefined threshold
14.    \( R\text{Start} = 0; \)
15.   If \( R\text{End} < \lambda \) then
16.    \( R\text{End} = n; \)
17.   Return \( R\text{Start} \text{ and } R\text{End}; \)
End

Fig. 6. The algorithm for removing noise.

of \( T_i \) and call \( gs_i \) the global similarity of \( T_i \). In Table 2, the global similarities of eight subtrees are shown in the last column. Obviously, the global similarities of \( T\text{review} \)s are much larger than those of \( T\text{noise} \)s.

In fact, the similarity of any two \( T\text{review} \)s is always between (3.5, 25), while the similarity range of noise subtrees is (0, 4.2). So it is unreasonable to select some value as the universal threshold to differentiate \( T\text{review} \)s and \( T\text{noise} \)s. Instead, we pay attention to the change rate of the global similarities, just like the curve shown in Fig. 5. We use \( gs_i / gs_{i-1} \) to represent the change rate of the global similarities, and \( n - 1 \) values can be obtained. If \( T\text{noise} \)s exist in \( T\text{region} \), the largest positive one must be the first \( T\text{review} \) and the \( T\text{noise} \) just ahead of it, and the smallest negative one must be the last \( T\text{review} \) and the \( T\text{noise} \) just behind it. Based on the consideration above, the algorithm for removing noise is depicted in Fig. 6.

The goal of this algorithm is to remove \( T\text{noise} \)s through finding the first \( T\text{review} \) and the last \( T\text{review} \). First, the similarities of any two child subtrees of \( T\text{region} \) are measured with the MTS algorithm and are stored in matrix \( T\text{SM}[n, n] \) (lines 1–8). Next, the global similarity of each subtree is obtained (lines 9–10), and the positions \( (R\text{Start} \text{ and } R\text{End}) \) of the first \( T\text{review} \) and the last \( T\text{review} \) are detected respectively (lines 11–12). In some review pages, no \( T\text{noise} \)s exists in \( T\text{region} \). To avoid the first and the last \( T\text{review} \)s being mistaken for noise subtress and being removed, we use the threshold \( \lambda \) to address such a situation; it is
4.3. Detecting boundaries of review records

After removing all T\text{noise}s, we need to detect the boundaries of the review records, i.e., find how many sequential T\text{review}s make up one review record (see DTF2); suppose that the number is \( l \ (l \geq 1) \). However, \( l \) is not easy to determine, due to the unpredicted amount of review records in a review page. Here, we introduce a simple and effective method to infer \( l \).

Based on the global similarity of T\text{review}s, a direct graph \( G(V, E) \) can be built: a vertex in \( V \) is one T\text{review}, and a direct edge in \( E \) exists from \( T_i \) to \( T_j \) if the global similarity of \( T_i \) is the similarity of \( T_i \) and \( T_j \). We call such a graph a global similarity relationship graph (GSRG for short). For each node in a GSRG, its outdegree must be 1, and its indegree is in the range \([0, n-1]\). For example, the GSRG for the case in Fig. 3 is the graph shown in Fig. 8 based on the global similarities in Table 1. In the GSRG, \( T_i \) and \( T_j \) have the same semantics and are from different review records if \( T_i \) points to \( T_j \), i.e., the \( i \)th T\text{review} of a review record must point to the \( i \)th T\text{review} of another review record. So the distance \( |i-j| \) is a multiple of \( l \). Motivated by this idea, the algorithm for detecting the boundaries of review records is shown in Fig. 7. The goal of this algorithm is to infer how many T\text{review}s a review record consists of; suppose this to be \( l \). First, the distance \( d_i = |i-j| \) is obtained for \( T_i \) based on the GSRG (lines 1–3), where \( T_i \) is the most similar T\text{review} with \( T_i \). Obviously, \( l \) must be a common divisor of \( \{d_1, \ldots, d_n\} \). If the maximal common divisor (suppose \( l' \)) is a prime number, it is returned as \( l \) (lines 5–6). Otherwise, it is possible for \( l \) to be \( l' \) or a divisor of \( l' \). Therefore the \( n \) T\text{review}s are segmented into \( n/l' \) subsequences averagely, and each sequence contains \( n/l' \) T\text{review}s. We recursively employ this algorithm until \( l' \) cannot become more smaller, and the current \( l' \) is returned as \( l \) (lines 8–10).

We use the example in Fig. 8 to illustrate this algorithm. According to the direct edges between the vertexes, the distances of \( \{T_2, \ldots, T_7\} \) are \( \{4, 2, 2, 4, 2\} \), which means that each review record contains at most two T\text{review}. So \( l \) is inferred to be 2. In real review pages, \( l \) is usually less than 6, and it can be inferred with at most one recursion in the algorithm for detecting the boundaries of review records.

5. Review content extraction

In this section, we discuss how to extract the pure review content from a review record. Due to the diversity of review contents, they are very inconsistent in terms of both tree structure and text length. The method for review content extraction is based on the inconsistencies of tree structure and text length.
5.1. Identifying the $T_{\text{review}}$ containing review content

If a review record consists of multiple $T_{\text{review}}$s, the $T_{\text{review}}$ containing the review content should be determined first. We group the ith $T_{\text{review}}$ of each review record together and get l groups: $\{G_1, \ldots, G_l\}$, where l is the number of $T_{\text{review}}$s that one review record consists of. The $T_{\text{review}}$s containing the review contents will be in the same group according to $D_T$. We use the standard deviation to select the group whose $T_{\text{review}}$s contain review contents. The formula for computing the standard deviation of each group is given as follows:

$$SD(G_i) = \sqrt{\frac{\sum_{j=1}^{N_i} (N_j - \overline{N} + 1)^2 (W_j - \overline{W})^2}{|G_i|}}$$

where $|G_i|$ is the number of $T_{\text{review}}$s in $G_i$, $N_j$ is the number of nodes of the jth $T_{\text{review}}$ in $G_i$, $\overline{N}$ is the average number of nodes of $T_{\text{review}}$s in $G_i$, $W_j$ is the text length of the jth $T_{\text{review}}$ in $G_i$, and $\overline{W}$ is the average text length contained in each $T_{\text{review}}$ in $G_i$. In the formula, we use $N_j - \overline{N} + 1$ instead of $N_j - \overline{N}$ to avoid the situation that all $T_{\text{review}}$s are isomorphic and only differ in text length. Obviously, the group with the largest standard deviation is the one whose $T_{\text{review}}$s contain the review contents.

5.2. Extracting the minimum subtree containing review content

The review content in a review record is one complicated subtree rather than one single node in the DOM tree. Therefore, the review content extraction process is actually to extract the minimum subtrees (denoted as $T_{\text{content}}$) that contain the pure review contents from the $T_{\text{review}}$s.

Among all semantics in review records, review content is the most inconsistent in terms of both tree structure and text length. Based on such considerations, we propose a smart three-step strategy for review content extraction: first, the $T_{\text{review}}$s containing review contents of different review records are aligned into one supertree (denoted as $T_A$); second, the consistency of each node in $T_A$ is measured, and the consistency of the subtrees in $T_A$ is measured based on the node consistency; third, $T_{\text{content}}$s are extracted from the review records. The technical details of the strategy are introduced as follows.

5.2.1. Tree alignment for reviews containing review contents

Given a set of trees, tree alignment is the process to construct a smallest common supertree $T_A$ or, say, that all the trees become isomorphic through inserting some nodes. Considering $T_{\text{review}}$ to be ordered label trees, we employ the fast tree alignment algorithm proposed in [10] to align two trees. The whole process of multiple tree alignment is described as follows. First, all $T_{\text{review}}$s are sorted in descending order according to their sizes. Next, the first two $T_{\text{review}}$s are aligned as $T'$, and then, the third $T_{\text{review}}$ and $T$ are aligned as $T''$. The process stops when all $T_{\text{review}}$s are aligned in turn, and $T_A$ is obtained. During the alignment process, the matching times of each node are logged. If the matching times of a node are m, this means the node appears in m $T_{\text{review}}$s. The upper bound of the matching times of a node is the total number of the aligned $T_{\text{review}}$s. Fig. 9 shows an example of tree alignment, and the numbers in brackets are the matching times of the nodes.

5.2.2. Node consistency and tree consistency

According to the tree alignment algorithm, the root nodes of $T_{\text{content}}$s must be aligned as one node in $T_A$, such as the root node of the subtree surrounded by the dashed cycle in Fig. 9. So review content extraction is accomplished if this node is detected. We still use $T_{\text{content}}$ to denote the minimum subtree in $T_A$ that contains the review contents if there is no confusion. We divide $T_A$ into two parts: $T_{\text{content}}$ and $T_A - T_{\text{content}}$, where $T_A - T_{\text{content}}$ denotes the remainder after $T_{\text{content}}$ is removed from $T_A$. Intuitively, there are two differences between the nodes in $T_{\text{content}}$ and those in $T_A - T_{\text{content}}$. First, the matching times of the nodes in $T_{\text{content}}$ are often smaller than those of the nodes in $T_A - T_{\text{content}}$. Second, if a node is in $T_{\text{content}}$, its text lengths in different review records are often different. To differentiate the nodes in $T_{\text{content}}$ or in $T_A - T_{\text{content}}$, we define the node consistency below:
Algorithm $T_{\text{content}}$ extraction

Input: $T_1, T_2, \ldots, T_l$ // $l$ the $T_{\text{review}}$s that are obtained by the method proposed in section 5.1
Output: $T_{\text{content}}$ // The root of the minimal tree that contains all review contents in $T_A$

Begin
1. $T_A = \text{TreeAlignment}(T_1, T_2, \ldots, T_l)$
2. For each node $a$ in $T_A$ do
3. Compute $\text{Consis}(a)$;
4. Initialize queue $Q$; // storing a path (from the root to a leaf node) of $T_A$
5. $a_r =$ the root of $T_A$;
6. Put $a_r$ into $Q$;
7. While $a_r$ is not a leaf node do
8. $a_c =$ the child node of $a_r$ whose $\text{Consis}(T_c)$ is minimum; // $T_c$ is $a_r$’s subtree
9. Put $a_c$ into $Q$
10. $a_b =$ $a_r$;
11. $s_{\text{max}} =$ $\infty$;
12. While $Q$ is not null do
13. $a_i =$ get a node from $Q$;
14. If $(\text{Consis}(T_A, T_i) - \text{Consis}(T)) < s_{\text{max}}$ then
15. $s_{\text{max}} =$ $(\text{Consis}(T_A, T_i) - \text{Consis}(T)); // T_i$ is $a_i$’s subtree
16. $a_m =$ $a_i$;
17. Return $a_m$
End

Fig. 10. The algorithm for review content extraction.

\[
\text{Consis}(a) = \frac{m}{n} \left( \sum_{i=1}^{m} \frac{|W_i|}{|W|} \ln \frac{|W_i|}{|W|} \right),
\]  

(3)

where $a$ is a node in $T_A$, $n$ is the total number of review records, $m$ is the matching times of $a$, and $|W_i|$ is the text length of $a$ in the $i$th review record; $|W|$ is the total text length of $a$ in all review records. Eq. (3) consists of two parts: $\frac{1}{n}$ and $\sum_{i=1}^{m} \frac{|W_i|}{|W|} \ln \frac{|W_i|}{|W|}$. The former is a measure of the consistency of $a$ on the tree structure, which is the frequency with which $N$ appears in review records. It is obvious that the consistency of a node is proportional to its matching number. The latter is to measure the consistency of $a$ on text length, which is based on the entropy concept in information theory. In information theory, entropy is a measure of the amount of disorder of a system. Due to the consistency of the nodes in $T_{\text{content}}$, we reasonably assume that the following equation is satisfied: $\text{Consis}(a_i) < \text{Consis}(a_j)$, where $a_i \in T_{\text{content}}$ and $a_j \in T_i - T_{\text{content}}$.

Based on node consistency, we further define the tree consistency as the average of the consistencies of the nodes in this tree, denoted as $\text{Consis}(T)$. According to the definition, the consistency of $T_{\text{content}}$ must be smaller than that of $T_i - T_{\text{content}}$, but it is not the minimum one because the consistencies of some subtrees of it are often more smaller. Thus it is not feasible to extract the subtree with the minimum consistency as $T_{\text{content}}$. Next, we will introduce the method to extract $T_{\text{content}}$ exactly from $T_A$.

5.2.3. Extracting $T_{\text{content}}$ from $T_A$

We first define the consistency difference as follows:

\[
\text{ConsisDiff}(T_i) = \text{Consis}(T_A - T_i) - \text{Consis}(T_i),
\]  

(4)

where $T_i$ is any subtree of $T_A$. Obviously, $\text{ConsisDiff}(T_{\text{content}})$ is a positive number because the nodes in $T_A - T_{\text{content}}$ are more consistent than those in $T_{\text{content}}$ on the whole. Actually, among all possible subtrees of $T_A$, $\text{ConsisDiff}(T_{\text{content}})$ is the maximum one. The reason is as follows. Assume there is another subtree $T'$ whose $\text{ConsisDiff}(T')$ is larger than $\text{ConsisDiff}(T_{\text{content}})$. There are three possible relationships between $T_{\text{content}}$ and $T'$: (1) $T'$ is a supertree of $T_{\text{content}}$; (2) $T'$ is a subtree of $T_{\text{content}}$; (3) there is no overlap between $T'$ and $T_{\text{content}}$. For the first possibility, the nodes in $T'$ do not belong to the review content, so $\text{Consis}(T_i - T_{\text{content}})$ is larger than $\text{Consis}(T_{\text{content}})$. We can conclude that $\text{Consis}(T_i)$ is larger than $\text{Consis}(T_{\text{content}})$, i.e., $\text{ConsisDiff}(T_i) < \text{ConsisDiff}(T_{\text{content}})$. For the second possibility, the nodes in $T_{\text{content}} - T_i$ belong to the review content, so $\text{Consis}(T_{\text{content}} - T_i)$ is smaller than $\text{Consis}(T_A - T_{\text{content}})$. We can conclude that $\text{Consis}(T_A - T_i)$ is smaller than $\text{Consis}(T_{\text{content}})$, i.e., $\text{ConsisDiff}(T_i) < \text{ConsisDiff}(T_{\text{content}})$. For the third possibility, it is obvious that $\text{Consis}(T_A - T_i)$ is smaller than $\text{Consis}(T_{A} - T_{\text{content}})$ because the nodes in $T_i$ do not belong to the review content, so $\text{Consis}(T_i)$ is larger than $\text{Consis}(T_{\text{content}})$. Based on this discussion, the algorithm for review content extraction is depicted in Fig. 10.

In this algorithm, the $T_{\text{review}}$s that contain review contents are aligned into $T_A$, and the consistencies of all nodes in $T_A$ are computed first (lines 1–3). Then, a path $Q$ is selected from using the following strategy (lines 4–10): the start of $Q$ is the root

Fig. 10. The algorithm for review content extraction.
Fig. 11. An example to illustrate the algorithm $T_{\text{content}}$ extraction.

Table 3

<table>
<thead>
<tr>
<th>Wrapper</th>
<th>Tag path</th>
<th>Position &amp; width</th>
<th>Font</th>
</tr>
</thead>
<tbody>
<tr>
<td>Review record</td>
<td>The tag path from the root of DOM tree to the root of $T_{\text{review}}$</td>
<td>The distance from the left border of the page to the left border of a review record and the width of a review record</td>
<td>The intersection of the fonts used in the review records</td>
</tr>
<tr>
<td>Review content</td>
<td>The tag path from the root of $T_{\text{review content}}$ containing review content to the root of $T_{\text{content}}$</td>
<td>The distance from the left border of a review record to the left border of its review content and the width of its review content</td>
<td>The intersection of the fonts used in review contents of the review records</td>
</tr>
</tbody>
</table>

of $T_A$, and a node is put into $Q$ if its parent node is also in $Q$ and the consistency of its corresponding subtree is minimum among its sibling subtrees. For each node $a_i$ in $Q$, $\text{Consis}(T_A - T_i) - \text{Consis}(T_i)$ is computed, and the node $a_m$ is as the root of $T_{\text{content}}$ if $\text{ConsisDiff}(T_m)$ is the maximum one (lines 11–17).

We use the example in Fig. 11 ($T_A$ in Fig. 9) to illustrate the algorithm. First, the root node $a$ is selected as the first node of path $Q$. For the child nodes $b$ and $c$, their consistencies are computed respectively, and node $c$ is selected if $\text{Consis}(c) > \text{Consis}(b)$. This process is repeated until the current node is a leaf node. Suppose that $Q$ is ‘$a \rightarrow c \rightarrow f \rightarrow i$’ (the path marked with red cycles in Fig. 11). Next, $\text{ConsisDiff}(T_a)$, $\text{ConsisDiff}(T_c)$, $\text{ConsisDiff}(T_f)$ and $\text{ConsisDiff}(T_i)$ are computed respectively. Node $c$ is selected as the root of $T_{\text{content}}$ if $\text{ConsisDiff}(T_c)$ is maximum among them.

6. Wrapper-based extraction

Given a review page, the review records and their review contents can be extracted respectively by employing direct extraction. The distinct advantage of this approach is independent of any web page template, but it also suffers from two limitations. First, the efficiency is low due to the time-consuming algorithms. The algorithms for tree matching and tree alignment are time-consuming when the sizes of the trees are large. Second, this approach will fail when there is only one review record in the page. In this section, we introduce another approach, wrapper-based extraction.

The wrapper-based extraction process generates the wrappers with one or more sample pages, and the review records and the review contents are extracted with the wrappers. In WeRE, only one sample page containing multiple review records is required. We sort the review pages according to their file sizes in descending order, and select the largest one as the sample page.

Wrapper generation

Two wrappers are generated, which are the review record wrapper and the review content wrapper. Each wrapper logs the features of the information it is extracting. The features can be classified into three types: tag path, position & width, and font. Table 3 gives the explanations of the three features in each wrapper.

We use one sample page to obtain these features. Direct extraction is employed to extract review records and review contents from the sample page, and get these features according to the extracted results.

Wrapper-based extraction

When the wrappers are generated, the review records and their review contents can be extracted in turn from the pages whose templates are similar to that of the sample page. We use review record extraction as an example to explain the extraction process. For review record extraction, the subtrees whose tag paths coincide with the tag path feature are extracted from DOM tree first. Because only the tag path feature cannot ensure that all of these subtrees are $T_{\text{review}}$, the noise subtrees are filtered according to the position & width feature and the font feature. The process of review content extraction is similar.
Table 4
The details on the dataset.

<table>
<thead>
<tr>
<th>Forum web site</th>
<th>The number of review pages</th>
<th>The number of review records</th>
</tr>
</thead>
<tbody>
<tr>
<td>forum.ubuntu.org.cn</td>
<td>1001</td>
<td>8,911</td>
</tr>
<tr>
<td>bbs.winos.cn</td>
<td>1001</td>
<td>8,118</td>
</tr>
<tr>
<td>wordpress.org.cn</td>
<td>1000</td>
<td>7,983</td>
</tr>
<tr>
<td>bbs.zdnet.com.cn</td>
<td>1000</td>
<td>1,800</td>
</tr>
<tr>
<td>bbs.tech.china.com</td>
<td>998</td>
<td>6,018</td>
</tr>
<tr>
<td>bbs.zhcw.com</td>
<td>999</td>
<td>8,172</td>
</tr>
<tr>
<td>bbs.chuguol.cn</td>
<td>999</td>
<td>5,780</td>
</tr>
<tr>
<td>hupub.com</td>
<td>1000</td>
<td>8,339</td>
</tr>
<tr>
<td>jd-bbs.com</td>
<td>998</td>
<td>12,173</td>
</tr>
<tr>
<td>bbs.55bbs.com</td>
<td>1014</td>
<td>8,411</td>
</tr>
<tr>
<td>bbs.imobile.com.cn</td>
<td>1000</td>
<td>17,186</td>
</tr>
<tr>
<td>sq.k12.com.cn/discuz</td>
<td>1000</td>
<td>16,726</td>
</tr>
<tr>
<td>bbs.czqzg.cn</td>
<td>999</td>
<td>3,343</td>
</tr>
<tr>
<td>forum.jiangmin.com</td>
<td>1001</td>
<td>9,127</td>
</tr>
<tr>
<td>bbs.ikaka.com</td>
<td>1001</td>
<td>6,703</td>
</tr>
<tr>
<td>itbbs.pcs.show.net</td>
<td>981</td>
<td>7,428</td>
</tr>
<tr>
<td>forum.livetome.cn</td>
<td>1000</td>
<td>4,302</td>
</tr>
<tr>
<td>bbs.mydrivers.com</td>
<td>1000</td>
<td>5,129</td>
</tr>
<tr>
<td>huatan.net/bbs</td>
<td>1000</td>
<td>8,594</td>
</tr>
<tr>
<td>bbs.canet.com.cn</td>
<td>1000</td>
<td>11,401</td>
</tr>
</tbody>
</table>

7. Experiments

This section gives a comprehensive evaluation for our system, WeRE (Web Reviews Extractor), which implements the proposed solution. Our experiments are carried out on an Intel Core 2 Duo 2.67 GHz CPU, 2G Memory PC. The data set and the evaluation metrics used in the experiments are presented first. Then, a series of experiments are reported and discussed.

7.1. Data set

The data set used in the experiments is the review pages crawled from 20 popular forum sites, and is denoted as FDS. We implemented a system to crawl review pages from the 20 sites. To ensure accuracy and efficiency, the crawler is semi-automatic. For each site, three features of review pages are predefined manually: string pattern of URLs, frequent key words of anchor texts, and the minimum number of hops from the root page of the site to the review page. In fact, only the first feature is site dependent, which means that this feature has to be defined for each site in turn. The other two features can be defined once and they are suitable for all sites. The review pages have been crawled in our local machines. The details of FDS are shown in Table 4.

7.2. Evaluation metrics and experiments settings

All previous works use precision and recall to evaluate their experimental results (some also include the $F$-measure, which is the weighted harmonic mean of precision and recall). These measures are also used in our evaluation.

WeRE provides two ways of web review extraction: direct extraction and wrapper-based extraction. No sample pages are required for direct extraction. For wrapper-based extraction, we select only one review page from each web site to generate wrappers. In addition, the threshold in review record extraction is set at 2.5.

7.3. Experiments on review record extraction

In this part, we evaluate the review record extraction of WeRE. The input is review pages, and the output is the extracted review records. We also compare it with two representative web data extraction systems, MDR [9] and ViDRE [4]. MDR mainly utilizes the DOM tree and extract data records directly, while ViDRE only utilizes the visual information of web pages and extracts data records with wrappers. Table 5 gives the experimental results for review record extraction, and Table 6 is the experimental results for MDR and ViDRE on our data set.

From Tables 5 and 6, two conclusions can be made. First, the performance of WeRE on review record extraction is close to perfect and is considerably better than that of MDR and ViDRE on all the three metrics. Both MDR and ViDRE perform web data record extraction under the assumption that the web data records are very similar with respect to the HTML tag or appearance, but the review contents impair the similarity seriously, while WeRE can weaken such impairment effectively. Second, in WeRE, the performances of wrapper-based extraction and direct extraction are close, but the recall of wrapper-based extraction is a little better than that of direct extraction. In several forum review pages, the first review record is too complicated compared to other review records to be mistaken as noise by direct extraction.
### Table 5
The experimental results on review record extraction.

<table>
<thead>
<tr>
<th>Method</th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct extraction</td>
<td>100%</td>
<td>98.9%</td>
<td>99.4%</td>
</tr>
<tr>
<td>Wrapper-based extraction</td>
<td>99.9%</td>
<td>99.0%</td>
<td>99.6%</td>
</tr>
</tbody>
</table>

### Table 6
The experimental results on MDR and ViDRE.

<table>
<thead>
<tr>
<th>Method</th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDR</td>
<td>64.6%</td>
<td>81.5%</td>
<td>71.9%</td>
</tr>
<tr>
<td>ViDRE</td>
<td>86.4%</td>
<td>82.2%</td>
<td>84.2%</td>
</tr>
</tbody>
</table>

### Table 7
The experimental results on review content.

<table>
<thead>
<tr>
<th>Method</th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct extraction</td>
<td>90.1%</td>
<td>87.3%</td>
<td>88.6%</td>
</tr>
<tr>
<td>Wrapper-based extraction</td>
<td>98.5%</td>
<td>98.3%</td>
<td>98.4%</td>
</tr>
</tbody>
</table>

### Table 8
Comparison results between WeRE and MLNs-PVV.

<table>
<thead>
<tr>
<th>Method</th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Review record</td>
<td>99.4%</td>
<td>99.1%</td>
<td>99.3%</td>
</tr>
<tr>
<td>Review content</td>
<td>97.3%</td>
<td>95.8%</td>
<td>96.5%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>WeRE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Review record</td>
<td>99.6%</td>
<td>94.1%</td>
<td>96.7%</td>
</tr>
<tr>
<td>Review content</td>
<td>91.4%</td>
<td>85.3%</td>
<td>88.2%</td>
</tr>
<tr>
<td>MLNs-PVV</td>
<td>94.0%</td>
<td>91.5%</td>
<td>92.7%</td>
</tr>
</tbody>
</table>

#### 7.4. Experiments on review content extraction

In this part, we evaluate the review content extraction of WeRE. The input is the extracted review records, and the output is the review contents extracted from these review records. A review content is considered as correctly extracted only if it does not contain any noise and does not miss any information. Table 7 shows the experimental results of the performances of review record extraction.

The experimental results shown in Table 8 indicate that WeRE can achieve very high performances on review content extraction. The performance of wrapper-based extraction is better than that of direct extraction, so we adopt wrapper-based extraction for review content extraction if the review pages come from one web site in real applications.

The approach proposed in [11] is similar to our work. 20 forum web sites (different from ours) are used in their experiments. Because no available system is provided by [11], we cannot make the performance compare between the approach proposed in [11] on our data set. Alternatively, 100 review pages are crawled from each web site provided in [11], and the experiments are carried out on this data set. The comparison results with the approach proposed in [11] (denoted as MLNs-PVV) are shown in Table 8.

As we can see from Table 8, the performance of WeRE is much better than that of MLNs-PVV. In particular, WeRE is more strict than MLNs-PVV in terms of review content extraction, while MLNs-PVV considers a review content correctly extracted if more than 95% content is contained.

#### 7.5. Experiments on efficiency

Besides accuracy, efficiency is another important factor in real applications. Therefore, we make an evaluation of efficiency for WeRE. All review pages have been downloaded in the local computer. Table 9 shows the experimental results.

The experimental results indicate that wrapper-based extraction is more efficient than direct extraction. On average, less than two pages can be processed by direct extraction in one second, while more than three pages can be processed by wrapper-based extraction. Wrapper-based extraction needs to generate three wrappers for each web site. The time cost of
wrapper generation is about 500 ms, but this can be ignored due to the large number of review pages of each web site. In addition, we also found that the main obstacle for speeding up the efficiency is the web page representation. In the current version of WeRE, we use the WebBrowser component in C# to parse the review pages into the DOM trees and obtain the visual information. Some unnecessary operations have to be performed during the parse process. In the future, we will choose an open-source system and simplify it to speed up the web page representation module.

8. Related work

Web review extraction belongs to the field of web data extraction. This field has received lots of attention in recent years. A good survey of current work on web data extraction can be found in [12]. According to the automation degree, all the works can be classified into three kinds: manual, semi-automatic, and automatic. Earlier works were mainly manual and semi-automatic [13–16]. Most current applications have to extract their desired data from a very large number of web pages. To improve the efficiency and reduce the manual efforts, most recent research focuses on automatic approaches instead of manual or semi-automatic approaches. In this section, we mainly introduce the automatic approaches.

Some representative automatic approaches are Omini [17], RoadRunner [18], IEPAD [19], MDR [9], and DEPTA [5]. Some of these approaches perform only data record extraction and not data item extraction, such as Omini. RoadRunner, IEPAD, MDR, DEPTA, and Omini do not generate wrappers, i.e., they identify patterns and perform extraction for each web page directly without using previously derived extraction rules. The techniques of these works have been discussed and compared in [12], and we do not discuss them any further here. In addition, there are several works (DeLa [20], DEPTA and the method in [21]) on data item extraction, which is a preparation step for holistic data annotation, i.e., assigning meaningful labels to data items. DeLa utilizes HTML tag information to construct a regular expression wrapper and extract data items into a table. Similar to DeLa, DEPTA also operates on HTML tag tree structures to first align data items in a pair of data records that can be matched with certainty. The remaining data items are then incrementally added. However, both data alignment techniques are mainly based on HTML tag tree structures, not visual information. The automatic data alignment method in [21] proposes a clustering approach to perform alignment based on five features of data items, including font of text. However, this approach is primarily text based and tag structure based, while our method utilizes the visual information on web pages as effective features. Recently, several works also utilize some visual information to extract web data, such as ViNTS [8], ViPERS [6], and HCRF [22]. ViNTS uses the visual content features on the query result pages to capture content regularities denoted as Content Lines, and then utilizes the HTML tag structures to combine them. ViPER also incorporates visual information on web pages for data records extraction with the help of a global multiple sequence alignment technique. Both of them only focus on data record extraction, without considering data item extraction. HCRF is a probabilistic model for both data record extraction and attribute labeling. Compared to our solution, the works above ideally assume that every data item corresponds to one leaf node in the DOM tree, but this does not hold for review records because their review contents actually correspond to inconsistent subtrees both in the DOM tree and on visual features.

Yang et al. [11] is the most similar work to ours. It incorporates both page-level and site-level knowledge to integrate much useful evidence by learning its importance. The site-level knowledge used by it includes (1) the linkages among different object pages, and (2) the interrelationships of pages belonging to the same object. However, it is a heavy burden to obtain site-level knowledge. In order to reconstruct the site map, a large number of non-review pages have to be crawled. Furthermore, a time-consuming clustering operation is needed for these crawled web pages. So this is impractical when the web pages are crawled from many web sites. Compared to it, our solution is lightweight: no complicated knowledge outside the inputted web pages is needed, and only one sample page is needed to generate the wrapper for each web site.

9. Conclusion

In this paper, we have provided a comprehensive solution to the problem of automatically extracting web reviews. We have tackled the challenges posed by user-generated review contents. The techniques have been described used in our solution to build WeRE. Our solution mainly consists of two steps. First, the review records are extracted from review pages by comparing the similarities of review records, and further, the review contents are extracted from review records by measuring the tree consistency. The experimental results on 20 forum sites show very encouraging performance and are superior to the existing solutions.
Acknowledgments

This work was supported in part by the China Postdoctoral Science Foundation-funded project under grant 20080440256 and 200902014, National High-tech R&D Program (2009AA011904 and 2008AA01Z421). The authors would also like to express their gratitude to the anonymous reviewers for providing some very helpful suggestions.

References


Further reading