Extracting Data Records from Web Using Suffix Tree

Xiaoqin Xie
College of Computer Science and Technology
Harbin Engineering University
150001, R.R.China
xiexiaoqin@hrbeu.edu.cn

Yixiang Fang
Department of Computer Science and Technology, Shenzhen Graduate School, Harbin Institute of Technology
518055, R.R.China
fangyixiang@hrbeu.edu.cn

Zhiqiang Zhang, Li Li
College of Computer Science and Technology
Harbin Engineering University
150001, R.R.China
{zqzhang,lily1989}@hrbeu.edu.cn

ABSTRACT
There are many automatic methods that can extract lists of objects from the Web, but they often fail to handle multi-type pages automatically. This paper introduces a new method for record extraction using suffix tree which can find the repeated sub-string. Our method transfers a distinct group of tag paths appearing repeatedly in the DOM tree of the Web document to a sequence of integers firstly, and then builds a suffix tree by using this sequence. Four refining filter rules are defined. After the refining processes we can capture the useful data region patterns which can be used to extract data records. Experiments on real data show that this method is applicable for various web pages and can achieve higher accuracy and better robustness than previous methods.

Categories and Subject Descriptors
H.2.8 [Database Applications]: Data Mining; H.3.5 [Online Information Services]: Web-based services

General Terms
Algorithms, Performance, Experimentation

Keywords
Data record extracting, suffix tree, unique tag path, pattern extracting

1. INTRODUCTION
Now Web-scale data is ever-growing and ever-changing. The Web also serves as a good user interface for databases available over the internet [1]. In order to effectively manage this Web-scale media, it is necessary to develop new automatic content extraction methods with higher accuracy to remove extraneous information or extract useful information from Web pages. Web data extracting becomes an important and challenging task in web data mining. A large amount of information on the Web is presented in regularly structured objects [3]. A list of such objects in a Web page often describes a list of similar items, which is also called data records. The objective of our work is to automatically mine all the data records in those HTML-based web sites. The data records are mainly associated with a fairly regular repeating structure which can be described as data region patterns.

Extracting the records from web pages can allow us to create extensive databases of entities for search engines to improve ranking and rendering of search results and to integrate web information. But, unsupervised or minimally supervised extraction techniques requiring little or no training do not have very high precision [2]. Furthermore, present record extracting methods such as MDR [3] often fail to handle more complicated or noisy web page structures due to their greedy manner of identifying a list of records through pairwise comparison of consecutive segments [1].

We have identified several key requirements for data records extracting methods from web pages: 1) Automaticity. The method ought to produce the detailed data records automatically. Any manual intervention will make the extract method infeasible because of the huge web scale data. 2) High precision. The method ought to make high precision information extraction at Web scale. 3) Noise immunity. In general, Web page is highly decorated. These decorative elements introduce more irregularity which is more harmful than helpful to data record extracting. These noise data can affect the quality of extraction. This phenomenon demands that the extracting method can identify these noise data and be robust to variations in site structure.

Present data extracting methods always fail to meet all above requirements. This paper introduces a suffix tree-based data record extracting method (STEM) which can meet all these requirements. STEM relies on a suffix tree to efficiently identify unique tag paths that form a data region pattern, to create repeating patterns and extract corresponding contents. One important difference between our approach and other related works is that we make no assumptions about the particular structure of a given webpage. STEM can be applied to most of the web pages.

Our main contributions can be summarized as follows:
1) Transform web page into a sequence of integers.
2) Propose a suffix tree model and define four refining filter rules to extract the data records template pattern.
3) Propose a new extraction technique which requires no training and have high precision and robustness.

This paper is organized as follows. Section 2 gives the problem formulation. Section 3 describes STEM in detail including suffix tree constructing, pattern refining and record extracting process. Subsequently, section 4 contains the experimental results. Section 5 describes the related works, and section 6 concludes the paper and gives a summary.

2. PROBLEM FORMULATION
In this section, we formally define the data extracting problem. Also, some observations on this problem are described.

2.1 Related Terms
One web page can contain multiple data regions and each data region can contain multiple data records of the same kind. The data records that constitute a Web page are typically represented using an HTML code template. Thus, they often have a similar appearance. Each data region is mainly associated with a
repeating html structure which we call data region pattern in our research.

**Definition 2.1 HTML Tag Path.** Based on the basic idea in [1], we view the Web page as a string of HTML tags. HTML Tag path is a path from the root to a leaf in the DOM tree of a Web page. This path can be presented as an ordered sequence of ancestor nodes in the DOM tree.

Therefore, each HTML tag can be mapped into a HTML tag path. Table 1 depicts the mapping from HTML tag to HTML tag path for one Web page example.

**Table 1.** HTML Code and HTML Tag Path

<table>
<thead>
<tr>
<th>HTML Code</th>
<th>HTML Tag Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;html&gt;</td>
<td>1 html</td>
</tr>
<tr>
<td>&lt;/html&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;body&gt;</td>
<td>2 html/body</td>
</tr>
<tr>
<td>&lt;/body&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;p&gt;My Web Page&lt;/p&gt;</td>
<td>3 html/body/p</td>
</tr>
<tr>
<td>&lt;/p&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;ul&gt;</td>
<td>4 html/body/p/ul</td>
</tr>
<tr>
<td>&lt;/ul&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;div&gt;Cell #1&lt;/div&gt;</td>
<td>5 html/body/p/ul/li</td>
</tr>
<tr>
<td>&lt;/div&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;li&gt;</td>
<td>6 html/body/p/ul/li/div</td>
</tr>
<tr>
<td>&lt;/li&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;div&gt;Cell #2&lt;/div&gt;</td>
<td>7 html/body/p/ul/li</td>
</tr>
<tr>
<td>&lt;/div&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;li&gt;</td>
<td>8 html/body/p/ul/li/div</td>
</tr>
<tr>
<td>&lt;/li&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;/ul&gt; &lt;/div&gt; &lt;/body&gt;</td>
<td>NA NA</td>
</tr>
</tbody>
</table>

**Definition 2.2 Unique Tag Path(UTP).** By traversing the DOM tree of a Web page’s html code, all the html tag paths can be gotten. The unique html tag paths can be identified from all the html tag paths in a web page. Each unique tag path is allocated a unique integer number. This number is called Unique Tag Path Identifier (UTPI). Table 2 lists all tag paths of Web page example in table 1. Table 3 lists the UTPs and UTPIs.

**Table 2. All tag paths and identifiers**

<table>
<thead>
<tr>
<th>All the tag path</th>
<th>identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>html</td>
<td>1</td>
</tr>
<tr>
<td>html/body</td>
<td>2</td>
</tr>
<tr>
<td>html/body/p</td>
<td>3</td>
</tr>
<tr>
<td>html/body/p/ul</td>
<td>4</td>
</tr>
<tr>
<td>html/body/p/ul/li</td>
<td>5</td>
</tr>
<tr>
<td>html/body/p/ul/li/div</td>
<td>6</td>
</tr>
<tr>
<td>html/body/p/ul/li/div</td>
<td>5</td>
</tr>
<tr>
<td>html/body/p/ul/li/div</td>
<td>6</td>
</tr>
</tbody>
</table>

**Table 3. Unique Tag paths and identifiers**

<table>
<thead>
<tr>
<th>Unique tag path</th>
<th>Identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>html</td>
<td>1</td>
</tr>
<tr>
<td>Html/body</td>
<td>2</td>
</tr>
<tr>
<td>Html/body/p</td>
<td>3</td>
</tr>
<tr>
<td>Html/body/p/ul</td>
<td>4</td>
</tr>
<tr>
<td>Html/body/p/ul/li</td>
<td>5</td>
</tr>
<tr>
<td>Html/body/p/ul/li/div</td>
<td>6</td>
</tr>
</tbody>
</table>

**Definition 2.3 Web Page Sequence.** The Web page can be presented as a sequence of all unique tag path identifiers. We call this sequence as Web Page Sequence. The number of the identifiers in this sequence is called the length of web page sequence. For example, the web page sequence is (1,2,3,4,5,6,5,6) in table 2, and its length is 8.

**Definition 2.4 Data Region Pattern.** The structured data in one data region of a Web page can be presented as a set of repeated html tag paths and each of paths will correspond to a data record. The repeated html tag paths can be considered as a data region pattern and we use the corresponding identifiers sequence to represent it. The number of identifiers in the data region pattern is called the pattern length. In a data region of web page, the html tag paths often appears consecutively.

For example, for the web page sequence which is (1,2,3,4,5,6,5,6), there is a data region and its data region pattern is (5,6). Its pattern length is 2.

**Definition 2.5 Visual Signal Vector.** Each unique tag path $P_i$ is related to a vector of binary visual signal $(S_i(1), S_i(2), \ldots, S_i(m))$, where $S_i(j) = 1$ means the UTPI of $P_i$ occurs in the web page sequence of HTML document at position $j$ and $S_i(j) = 0$ otherwise, and $m$ means the length of the Web page sequence. Figure 1 gives an example of visual signal vector about example in Table 1.

**Figure 1.** An example of visual signal vector.

Thus one web page can be converted into the visual signal vectors. By using bright pixels to stand for ‘1’s and dark pixels to stand for ‘0’s, we can present a Web page to a visual view. For example, Figure 2 gives the visual representation of all the unique tag paths of web page in Table 1. Each row is a visual signal vector.

**Definition 2.6 Web Page Suffix Tree(WPST).** Each integer of the web page sequence can be regarded as one or more characters, so the sequence can be regarded as a string. The Web page suffix tree is a compact trie containing all the suffix sequences of the web page sequence. The Web page suffix tree is a rooted, directed tree. Each internal node has at least 2 children. Edges are labeled by strings. No two edges out of the same node can have edge labels that begin with the same word (hence it is compact). For each suffix $suf$ of the web page sequence, there exists a suffix node whose label equals $suf$. The suffix tree of a collection of web page sequences is a compact trie containing all the suffixes of all the web page sequences in the collection.

**Figure 2.** Visual representation of the example in Table 1.

**Definition 2.7 Node Sequence.** For each node in the web page suffix tree, we traverse from root node to it and concatenate the edge labels on the path. Thus we can get a subsequence for each
node which is called node sequence. Notice that the node sequence is the sub-string of the original web page sequence. The node sequence is also the label of the node in the web page suffix tree. In addition, each node has a repeating time property which gives the repeating times of the node sequence string occurring in web page sequence. As depicted in figure 3, the number in each circle is the repeating time of the node sequence.

For example, the web page sequence is (1, 2, 3, 4, 5, 6, 5, 6). For the node P, traversing from root to node P, we get its node sequence (5, 6, 5, 6) and it appears in web page sequence only once, so we mark the circle of P node with number “1”. Now, the goal to find the data region patterns can be transferred into finding the repeated subsequence of the web page sequence. The suffix sequences of this web page sequence (1, 2, 3, 4, 5, 6, 5, 6) are (1, 2, 3, 4, 5, 6, 5, 6), (2, 3, 4, 5, 6, 5, 6), (3, 4, 5, 6, 5, 6), (4, 5, 6, 5, 6), (5, 6, 5, 6), (6, 5, 6), (5, 6), (6). By using STEM, we can find that there is only one data region and the data region pattern is (5, 6) in this example.

Notice that, by default, in STEM each sub-string of the original web page sequence is gotten by traversing suffix tree from root node to any other nodes, that is to say from top to down. When traversing via bottom up approach, the sub-string achieved may not be the sub-string of web page sequence.

Problem Definition. Given a web page html f, we can get the corresponding web page sequence wps and construct the WPST st. Then the problem of discovering the potential data region pattern drp can be transferred to the problem of discovering the repeated sub-sequences rst in the st.

2.2 Observations
A data region is part of a Web page that contains multiple data records of the same html tags which can be consecutive or non-consecutive [1]. We observe that the following features are always available in the data region of most Web pages.
1) One Web page may contain multiple data regions, but the number of data regions is limited (such as less than 10).
2) Any two different data regions in one web page are non-intersecting.
3) Each data region contains more than two data records. The number of repeated tag paths reflects the number of data records in one data region.
4) Each data record contains at least two attributes. This means that the length of the corresponding sub-sequence is at least two. The length of data region pattern is finite. In our research, we suppose the length is less than forty.
5) Little or even no noise data exists between data records.

3. STEM METHOD
This section describes the Suffix Tree-based data record Extracting Method (STEM), including the algorithm steps and the refining filters in detail.

3.1 Basic Idea
Assume that the web page html is f. After data cleaning and preprocessing f, we can get the corresponding web page sequence wps. Then the web page suffix tree st of wps can be constructed. By traversing st and refining the results, the set of repeated sub-sequences Rst can be found. After filtering those noise sub-sequences, the optimal sub-sequence rsts are just the data region patterns that we want. STEM has five steps which will be described in detail in section 3.2. The STEM algorithm is summarized in the following:

STEM Algorithm
Input: Web page’s HTML file f ;
Output: Date records set R in the HTML file f.
1. Preprocessing the original web page file f;
2. Obtaining web page sequence wps;
3. Building the web page suffix tree st using wps;
4. Traversing st, refining the data region pattern set drps, and choosing the optimal pattern drp;
5. Extracting data records R by using data region pattern drp.

3.2 Algorithm Main Steps
Step1: Preprocessing the original web page file f.
As we all known, the html source code of web page is usually badly organized. Some html tag may lose close tags and some may match other html close tag. So it will be difficult for later data mining process if we don’t conduct any data cleaning or preprocessing. Data preprocessing is one of the important steps, which involves these tasks including html formatting and filtering.

Since the first aim of extracting data records from web pages is to extract the repeated patterns, we focus on the html tags which have a great effect on the web page’s structure. We use Jtidy\(^1\) tool to repair the original page’s source code to a very pretty and normative format. Those nodes such as em, b, u, i and font, etc. which have little effect on the structure are filtered.

Step2: Obtaining the web page sequence wps.
By traversing the DOM tree of the web page’s html code, we can get the html tag path of each html tag easily. From all the html tag paths, we filter the repeated html tag paths and obtain the set of unique tag paths, and then we assign an integer identifier for each unique tag path. Based on all the identifiers of the unique tag paths of the original web page, we get an integer sequence which is the web page sequence wps. Each integer of the sequence can be regarded as one or more characters, so the sequence can be regarded as a string.

Note that, when assigning the integer identifier, in general the earlier the node appears when traversing the DOM tree, the smaller the integer identifier which is assigned to the node’s corresponding unique tag path is.

Step3: Building the web page suffix tree st.
The web page sequence wps is used to build the suffix tree st. We use the classical O(n) suffix tree constructing algorithm[4]. By traversing the suffix tree, we count the repeating times for each node and label these values to the nodes of st. Each node in st can be considered as an original candidate data region pattern.

For each node x in the st, we traverse st from root node to x, accumulate the labels of those edges on the traversing path, and then we get the node sequence ns for node x.

Step4: Refining the data region pattern.
We now identify each data region by finding repeating node sequence in suffix tree. Based on our observations, the data records in one data region often appear consecutively and these

\(^{1}\) Jtidy: http://sourceforge.net/projects/jtidy/?source=directory
data records will not appear in other data regions, so we can filter some patterns which are too short. For each candidate pattern, we can find its appearing position in the web page sequence and its repeat times. We find that there are some special cases in which the repeating times are different between the results created from the suffix tree traversing and that in practical. For example, there is a node sequence (1,2,1,2,1,2) and its subsequence is (1,2,1,2). This subsequence repeats twice consecutively. However, when traversing the suffix tree and counting the repeating times of the subsequence we will get three. In this example, the real repeating time of this pattern is 2 instead of 3. So it is necessary to refining the candidate data region patterns.

At this step, the first task is to get the frequency of each suffix tree node which refers to a sub sequence. The second task is to choose those optimal subsequences that agree with the feature of data region. We use four refining functions including node filter, gap filter, repeating filter and tree filter to select our target patterns from the candidate data region patterns. The detailed description is referred to section 3.3.

Step 5: Extracting data records in terms of extracted patterns.
After step 4, we get the data region patterns. For each pattern, we can obtain a group of html tag paths. Using these tag paths, we can easily locate the start and end position of each data region in a web page.

We extract data records one by one. We firstly extract the data records which match the pattern completely. In fact, some patterns which do not match the pattern completely may be what we need, due to the fact that people's tolerance of some noise data. For sections of consecutive html tag paths which match the pattern incompletely, we define and calculate a Usability Probability for them. Assume the data region pattern is ns (in fact ns is a node sequence), the web page sequence is wps, the ns occurs in wps at multiple positions (p1, p2, ..., pK). So wps can be presented as (os0, ns, os1, ns, os2, ..., osK ns, osK+1). os is a string which can be null. For a data region pattern ns, the usability probability of the noise data os with non-null value can be calculated as follows:

\[ \text{UsabilityProbability}(os) = \frac{L}{\max(N, X)} \]  \hspace{1cm} (1)

Where L is the length of the largest common subsequence between os and ns, N is the length of ns and X is the length of os.

If the value of UsabilityProbability for a noise data is large enough, it means that the noise data can be tolerated and we should extract it, or else we ignore it. In our research we choose 0.5 as the threshold value and it performs very well.

3.3 Refining Functions
In STEM, we define four refining functions. When traversing the suffix tree, the using order of the four refining function is node filter, gap filter, repeating filter and tree filter sequentially.

1) Node Filter: As we have observed, each data region has at least two data records. Each record has at least two different html tags and the length of the data region pattern has a maximum threshold. So the corresponding data region pattern must repeat at least twice and has at least two different html tag path identifiers. The data region pattern length will not be very long also. Therefore, when traversing the web page suffix tree, we prune those nodes which meet the following requirements: 1) Repeat only once. 2) The node sequence contains only one unique integer. 3) The node sequence is larger than the threshold value.

In our experiments, we choose 40 as the upper threshold value of the pattern length. Our experiments have shown that the number of data regions of most web pages would be less than 10. So after node filter pruning, we get those nodes that possibly include the potential data region pattern. We regard the node sequences of these nodes as the candidate data region patterns. Each pattern corresponds to a group of html tag paths.

Figure 4 gives an example of node filter pruning. Those nodes marked a cross are pruned because their node sequences just contain one character or the repeating time is only 1.

2) Gap Filter: For each candidate data region pattern, we can locate its positions in web page sequence easily. Each data region pattern corresponds to a node sequence. The node sequence will occur more than once in web page sequence at different positions. Gap filter function will check whether the same node sequence occurs in web page sequence consecutively. If not consecutive, the corresponding node will be pruned. For all the occurrence positions of the node sequence, if they appear non-consecutively, that is to say, if the ending position of the former isn’t next directly to the starting position of the latter, it means there maybe exist other string between the two matched node sequence in web page sequence and these string can be noise data in this data region.

For example, assume a web page sequence is (1,2,3,4,2,3,4) and the data region pattern is (2,3,4). (2,3,4) occurs in (1,2,3,4,2,3,4) at position 2 and 5. When calling gap filter function, the first matching position is starting from 2 and ending at 4, the second matching position is starting from 5 and ending at 7. 5 is next to 4, so the data region pattern agree with the gap filter rule and the corresponding node will not be pruned.

In order to filter the very short pattern, we define the valid rate measurement to evaluate the extent to which the data region pattern is disturbed by noise data, which can be calculated as follows:

\[ \text{ValidRate} = \frac{N}{N + M} \]  \hspace{1cm} (2)

Where N is the number of data region pattern occurs in the web page sequence, M is the number of invalid substring between first and last positions that data region pattern occurs in web page sequence. We choose 0.75 as the valid rate in our experiments. If the ValidRate is too low, we can prune the corresponding node and the data region pattern will be ignored, because there are many mismatched string among the matched subsequence. And it also means that there are many noises in this data region.
3) Repeating Filter: After the above two filtering steps, we find that there are patterns which are the repetition of other patterns. We call these patterns as repeating pattern. For example, assuming one data region pattern is (1,2,3,1,2,3), we can easily understand that the real pattern should be (1,2,3), rather than (1,2,3,1,2,3). But when we traverse the suffix tree, this pattern (1,2,3,1,2,3) must also appear in candidate pattern set, so we filter this kind of patterns. For each candidate pattern, if it is just a several repetition of another pattern, we filter it.

\[ \text{Figure 5. Tree Filter Example.} \]

4) Tree Filter: For each candidate pattern, the order of the node sequence may be rotated. The suffix tree will result in a wrong pattern. For example, the node sequence is (1,2,3,4,2,3,4,2), so the pattern should be (2,3,4). But when traversing the suffix tree shown as figure 5, the pattern (3,4,2) which is the node sequence of the node in the suffix tree will be yielded. Aiming to solve such problem, we define the tree filtering rules.

Rule 1: for each node sequence, if its corresponding html tag path is the true substring of the first identifier’s corresponding html tag path, it means that the pattern needs rotation. For example, assume the node sequence is (2,3), and the node sequence’s corresponding html tag paths are "/table/tr/td " and "/table/tr". Because "/table/tr/td" is the sub string of “/table/tr/td”, so we rotate (2,3) to (3,2).

Rule 2: if the html tag path corresponding to the ith (i>=2) identifier character of the node sequence isn’t the true substring of the first identifier’s html tag path, it means that the two tag path are parallel. If the former integer character is bigger than the latter one, it means that the pattern need rotation, otherwise, it need not.

In this refining process, we first check whether the pattern properly matches the above two rules. If it does not, we first move the last one integer identifier to its head position to get a new sequence, and check whether it meet the above two rules. If it does after this rotation movement the refining process terminate, otherwise we move the last integers to its head position and check it again in terms of the same method. For example, the node sequence is (1,2,3,4,5,6,7), then if need rotating, after moving 7 to the first position we get the corresponding new sequences are (7,1,2,3,4,5,6) , (6,7,1,2,3,4,5), (5,6,7,1,2,3,4) and so on. This check process will repeat until the sequence is qualified for above two rules.

After the four refining filter process, we can obtain patterns whose length are reasonable and theirs orders are proper. Those patterns are the target data region patterns.

3.4 Algorithm Discussions

Assume the length of web page sequence is n. The step for obtaining web page sequence takes O(n) time. The step for constructing the suffix tree takes O(n) time. As there are less than 10 data regions in a web page, which also means that there are less than 10 patterns needed to be detected in the suffix tree, so the cost of the four filter function will be constant. And the step for extracting the records is O(n). So the overall time complexity of STEM algorithm is O(n).

One of advantages of STEM is that it permits the occurrence of the noise data. And STEM can find any potential data region in theory.

4. EXPERIMENTS

We conducted extensive experiments on different real data set to evaluate the effectiveness and efficiency of the STEM algorithms. We compare our STEM with MDR[3] and TPC[1] approaches.

4.1 Data Set

Data set 1 was chosen from the testbed data extraction from the deep web, collected by Yamada et al. [5]. The testbed data has 253 web pages from 51 web sites randomly drawn from 114,540 web pages [1].

Data set 2 was the returning result pages of 15 different search engines such as google, baidu and so on by using the query keyword “Einstein”. We take the top five original web pages as our experiments dataset and there are 15*5=75 pages. Generally speaking, the first page of the returning result contains plenty of noise data.

Data set 3 was the returning result pages from other 15 different websites by using the query keyword “Einstein”, with the range of domains including multi-media, news, and shopping, such as play.google, ebay, amazon and so on. We take the top five original web pages as our experiments dataset and there are 15*5=75 pages in this data set.

One data region has multiple data records. Each data record has multiple data fields. Each data field has texts with different length. So the summary length of all data field texts of all data records in one data region is called the length of the data region. For those pages with multiple data regions, we extract all the data records of different data regions, and choose the data region with max length as the result data for comparing.

4.2 Experiment Setup

All methods were implemented in Java and conducted on an Intel Core computer with a 2.53GHz CPU, 4GB of RAM and Window 7 OS. As a matter of convenience, we abbreviate the method in [1] as TPC. For verify the effectiveness and efficiency of the STEM, we compare the performance of our STEM with that of MDR and TPC approaches when extracting data region pattern and data records. And we analyze the results in following three aspects:

1) Data region pattern similarity comparison. By regarding each integer in the data region pattern as a character, we can calculate the edit distance between the standard pattern sp and the extracted pattern ep. Let L, M denote the length of the standard pattern and the edit distance between patterns. Therefore, we can define the pattern similarity \( \text{patternSim}(sp, ep) = \frac{(L-M)}{L} \) as follows.

\[
\text{patternSim}(sp, ep) = \frac{(L-M)}{L}
\]  

We evaluate the data region pattern mining performance of different approaches in terms of pattern similarity. If the \( \text{patternSim} \) value of extracted data region pattern is lower than the threshold value ts, it means that the pattern is wrong or too incomplete. In our experiment, we set ts as 0.5.
2) Data record extraction accuracy comparison. We evaluate the data record extraction performance of different approaches in terms of Precision (Prec.) and Recall (Rec.) on three data sets. We report the extracting precision and recall rate of MDR, TPC and STEM approaches for the different test cases respectively.

3) Robustness performance comparison for noise data. We use the data record extraction performance results to evaluate the robustness performance comparison for noise data.

4.3 Results

4.3.1 Data Region Pattern Similarity Comparison

In order to compare the performance of STEM and TPC for data region pattern extraction, we use the first returning page of the website in data set 2 and data set 3 for our experiment and it contains 30 web pages. Figure 6 shows the pattern similarity of STEM and TPC for 30 web pages. We see that, our method clearly has much higher similarity for most web pages. For more that half of pages, the pattern similarity for TPC is less than 0.50. For our STEM, there are only three pages whose pattern similarities are less than 0.50 and most of the pattern similarities are larger than 0.75. This result shows that the data region patterns extracted by STEM are more complete and useful than that by TPC.

![Figure 6. Pattern similarity comparison between TPC and STEM for 30 web sites.](image)

**Figure 6.** Pattern similarity comparison between TPC and STEM for 30 web sites.

In addition, for data set 2, we found that TPC fail to extract data patterns from the first pages returning from ASK, Baidu, Bing and Yahoo search engines et al., while our STEM can extract such kind of patterns. Figure 7 depicts the visual signal vector with noise data of the first returning page from “Baidu” search engine to the keywords “Einstein”. The rectangle circles the data region pattern. The eclipse circles the noise data around the real data region. Our STEM can extract such kind of page template pattern. Figure 8 shows the data region pattern extracted by STEM for first page returning from “Baidu”. And the data region pattern can be represented as (22,23,24,31,32,38,33) which is also the standard pattern. But the pattern extracted by TPC is (23,31,32). By analysis, we find that the reason for the TPC method’s failure is that the TPC use a spectral clustering algorithm to cluster the pairwise similarity matrix of unique tag path. And it will perform badly when the similarities of the input matrix are too low, due to the existence of the noise data. It means that the data region can’t be detected by clustering algorithm. The same phenomenon exists for other result pages returning from search engines as Ask, Bing and Yahoo et al.

4.3.2 Data Record Extraction Accuracy Comparison

Some result data of MDR and TPC in data set 1 is adopted in [1] directly. As depicted in table 4, for data set 1, the precision and recall rate of STEM is higher than that of TPC and MDR. We see that our STEM improves the performance than MDR and TPC(for example, about 30% and 1% by Precision than MDR and TPC).

![Table 4. Data record extraction accuracy of different methods in data set 1.](image)

<table>
<thead>
<tr>
<th>Data</th>
<th>Algorithm</th>
<th>Prec.</th>
<th>Rec.</th>
</tr>
</thead>
<tbody>
<tr>
<td>data set 1</td>
<td>MDR</td>
<td>0.6793</td>
<td>0.6409</td>
</tr>
<tr>
<td></td>
<td>TPC</td>
<td>0.9684</td>
<td>0.9589</td>
</tr>
<tr>
<td></td>
<td>STEM</td>
<td>0.9698</td>
<td>0.9600</td>
</tr>
</tbody>
</table>

If the patternSim value of extracted data region pattern is lower than the threshold value, it means that the pattern is wrong or too incomplete. So we don’t use it for record extraction.

For data set 2, we have extracted the data pattern from the top one page to top five pages respectively. Each page has about 10 data records averagely. Figure 9 and 10 depict the performance of TPC and STEM for data set 2 and 3. We see that our STEM shows better performance than TPC. For example, in figure 9, for data set 2’s first page, our STEM clearly improves the performance about 65% by precision and 48% by recall.

![Figure 9. Robustness performance comparison between TPC and STEM for data set 2.](image)

![Figure 10. Robustness performance comparison between TPC and STEM for data set 3.](image)
As shown in figure 9, for the second and next page of each web site, the average precision and recall become stable. We can find that the data records in the result pages yielded from search engine are consecutive. Once the template pattern is identified, it is generally that the data records can be extracted correctly. For those web pages with strong noise such as from search engines, since the similarity is very small, it is often too hard for TPC method to get the data pattern. So the first page data for TPC in figure 9 is very low (about 32% on precision and 30% on recall). In addition, the pattern extracted by TPC is often incomplete and this often leads to the incorrect data record extraction results.

4.3.3 Robustness Performance Comparison
In this experiment, we use the data record extraction precision and recall measurement to evaluate the robustness performance for noise data, and we focus on the data set 2 which came from the returned pages from search engines and data set 3 which came from different domains including multi-media, news and e-shopping.

![Figure 11. Accuracy comparison between TPC and STEM for data set 2](image)

**Figure 11.** Accuracy comparison between TPC and STEM for data set 2

![Figure 12. Accuracy comparison between TPC and STEM for data set 3](image)

**Figure 12.** Accuracy comparison between TPC and STEM for data set 3

We found that the data set 2 (including pages returned from search engines) has the following properties: 1) The noise data mainly exists in first page. 2) The second and next page’s records often have the same structure, but they have much less noise data compared with the first page. So in this experiment, we focus on the first page returned from the different search engines. That is to say that we focus on about 10 data records extracting results. The experimental results on data set 2 for our STEM algorithm compared with TPC are shown in figure 7. For data set 3, each of the five pages often has the similar structure. The noise data scatters in each page. So we just list the first page results as shown in Figure 8. In figure 7 and 8, the ground truth is the set of data records in first page from one Web site. True positives are the set of data records correctly extracted by the algorithms from that Web site. False positives are the set of data records that the algorithm incorrectly includes in the same list with the true positives. As shown in figure 7 and 8, we can see that STEM has more true positives and less false positives compared with the TPC methods.

5. RELATED WORKS
There exist many approaches to extract structure data from HTML pages. Early work on wrapper induction utilizes manually labeled data to learn data extraction rules. But such semi-automatic methods are not scalable enough for web scale data. P. Gulhane et al. [2] developed a wrapper induction system called Vertex for extracting structured records from template-based web pages. Vertex focuses on the high-precision information extraction at web scale. The system needs human editors to annotate the attribute values on a few sample pages belonging to each Web site annotations. But how much the amounts of human effort are enough is an uncertain problem. The more the human effort are, the higher precision the extracting result will have. In contrast, our suffix tree algorithm need not learning process and is more robust for diverse web pages. Furthermore, we can see that from the experimental results, our STEM is fitful to process such pages yielded from search engines. Zheng et al. [6] proposed wrappers with a “broom” structure to extract records with interlaced attribute values within a page. But their method is not resilient to tag modifications. Etzioni et al. [7] exploit linguistic patterns and redundancy in the Web data for extraction. But this approach has low precision.

Most automatic technologies for web record extraction employ a particular similarity measure between web page segments to identify a region in the page where a similar data object or record appears repeatedly. For example, MDR [3] uses the edit distance between data segments. But MDR fails as the web page structure becomes more complicated [3][1]. Mohammed Kayed [8] proposed a new algorithm for multiple string alignment. Each row in the matrix represents one string of characters, where every character corresponds to a subtree in the DOM tree of a web page. Two subtrees take the same symbol in the peer matrix if they are similar, where similarity can be measured using either structural, content, or visual information. Miao et al. [1] presented a record extracting method based on tag path clustering (we abbreviate it as TPC), which focuses on comparing the pair of tag path occurrence patterns (called visual signals), and introduced a similarity measure that captures how closely the visual signals appear and interleave. Miao et al. view the web page as a string of HTML tags. TPC clusters these tag paths based on this similarity measure and extracts sets of tag paths that form the structure of the data records. In our research, we adopt the idea of visual signals. But instead of viewing the tag path as a unique visual pattern, we label each unique tag path as one integer, and a web page can be considered as a sequence of integer. A suffix tree is constructed based on the sequence. So our method need not to cluster the unique tag path, instead we build a suffix tree based on the unique tag path to find the repeated patterns. Thus our method is more robust and resilient to the noisy data. And our STEM method is fully automatic and does not involve human labelling or feedback. In addition, there are following drawbacks in the TPC method using tag path in [1]. First, it is very difficult to differentiate those data region when the tag path of non data area and data area are same. Our STEM method can tolerate this case. Second, TPC need to adjust some key coefficients and this task is difficult, where there are little coefficients in our method which are easy to control and not critical to final results. Third, TPC focuses on extracting a set of data records from a single page [1], while STEM can extract the underlying template patterns from multiple pages. In STEM, one suffix tree can be constructed for all the sequence of multiple pages. Thus all the template patterns from multiple pages can be detected.
There are many efficient algorithms for mining frequent sequences[12,13,14]. And these mining techniques have applied to special data types such as the Web, music and biological. Compared with the existing frequent sequence mining methods, our STEM can detect all frequent sequence. But that is just a step in STEM. The final goal of STEM is to detect the data region in Web page and discovery the data region pattern. STEM is more specialized than those enforced in frequent sequence mining methods. So four refining filter process in STEM is the critical parts which other methods do not have.

The suffix tree is one of most widely adopted indexes in the application of genome sequence alignment [14]. The idea of using a suffix tree for document clustering was introduced in [9]. Hung Chim et al. [10] proposed a similarity measure to compute the pairwise similarity of text-based documents based on suffix tree document model. Bing et al. [11] utilize suffix tree structure to identify the tandem repeats. They focus on the two sub-tasks of record region detection and record segmentation in a unified manner. They design a search structure named as RST. Based on the RST, they propose a token-based edit distance which takes each DOM node as a basic and inseparable unit in the cost calculation and develop several efficient search pruning strategies on the RST structure of a given region to identify the correct record segmentation. Our STEM uses the suffix tree model to find the repeating patterns.

Compared with other approaches, our STEM has the following advantages:
1) The time complexity is low. STEM takes \( O(L) \) time, while the time complexity of the TPC algorithm is \( O(M^2 + L) + O(M^2) \), where \( L \) is the total number of tag occurrences and \( M \) is the number of unique tag paths in the Web page.
2) STEM does not need training and learning process. STEM doesn't use clustering techniques. Besides, its parameters are not sensitive to different cases and they can be set easily by users.
3) Data region pattern Extracted by STEM is more complete and ordered than that by TPC. Such pattern is qualified to extract data records. But the pattern gotten by TPC is often disordered and needed to select the root node.
4) STEM can tolerate more noise data and this also means that STEM is more robust than TPC. TPC often fails to extract patterns from pages with plenty of noises, for its using clustering technique whose result is uncertain. As for STEM, STEM can extract all the patterns as long as there are repeated patterns in web pages. Therefore, STEM can be used to various web pages.

6. CONCLUSIONS
This paper presents a novel approach STEM for extracting data records from Web pages. STEM is an extraction technique which requires no training and has high precision and high robustness to noise data. In our method, we extract html tag paths from web pages firstly and transfer those paths to a sequence of identifiers. Secondly, we construct a suffix tree by using this web page sequence and detect data region patterns from the suffix tree by four refining filter functions. After four refining processing, we can filter the incorrect candidate data region patterns and obtain the target patterns. Compared with previous methods such as MDR, TPC etc., our STEM method performs more efficient which is also proved by the experiments. Especially, our method performs much better than those previous methods for Web pages which contain plenty of noise. This also means that our method is more practical. Our work has presented a successful approach to find the data region patterns and extract data records without learning and training process. We are exploring ways of more efficient refining algorithm to boost extraction accuracy.

7. ACKNOWLEDGMENTS
This research of data records extracting method from Web using suffix tree is supported by the Natural Science Foundation of HeiLongJiang Province under grant No. F201016 and F201130, and the Science and Technology Innovation Talents Special Fund of Harbin under grant No. RC2010QNO10024.

8. REFERENCES