Hybrid Method for
Automated News Content Extraction from the Web

Yu LI[1] , Xiaofeng Meng[1], Qing Li[2], and Liping Wang[2]

[1] School of Information, Renmin University of China, China
{liyu17, xfmeng}@ruc.edu.cn
[2] Dept. of Computer Science, City University of Hong Kong, HKSAR, China
{50095373@student, itqli@}.cityu.edu.hk

Abstract. Web news content extraction is vital to improve news indexing and searching in nowadays search engines, especially for the news searching service. In this paper we study the Web news content extraction problem and propose an automated extraction algorithm for it. Our method is a hybrid one taking the advantage of both sequence matching and tree matching techniques. We propose *TSReC*, a variant of tag sequence representation suitable for both sequence matching and tree matching, along with an associated algorithm for automated Web news content extraction. By implementing a prototype system for Web news content extraction, the empirical evaluation is conducted and the result shows that our method is highly effective and efficient.

1 Introduction

WWW has posed itself as the largest data repository ever available in the history of humankind, and Web search engines such as Google, Yahoo, and MSN Search have emerged as the most important Web services in recent years. Besides conventional keyword based and general purpose search, most search engines have recently launched a new searching service, named "Web news search", which mainly focuses on Web pages of news.

Web news search causes some non-trivial problems to traditional information retrieval techniques. One of them is how to differentiate Web news content from others in Web pages. As we know, a Web news page usually contains not only the content (the title, date and context) of news, but also other facilities such as the navigation area of the whole Web site, links to related topics, links to supplemental materials, advertisements, and even ongoing events. Separating the news content from the others, and only indexing the content, are believed to be the way to further improve the searching quality.

In this paper we focus on automatically differentiating Web news content from others. In general, identifying the content of an article in conventional techniques requires the knowledge on semantics of phases, which is a hard problem in itself. However, in Web news scenario, alternative solutions are possible, because Web pages are semi-structured documents written in markup language (mostly in HTML), which contain tags and structures dividing the context into fragments with specific

semantics. By properly utilizing them, recently proposed techniques to reduce patterns or to match similar structures, reported can be applied to automatically extract specific parts of HTML documents. Therefore we are motivated to find a similar solution to automatically extract Web news content.

According to the Web page modeling method, existing Web extraction techniques generally fall into two main categories: tag sequence based v.s. tree based. Techniques of the former category view a Web page as a long sequence consisting of HTML tags and text fragments. And the basic idea is to apply traditional pattern reduction techniques in order to find a template. After continuous improvement, these techniques can be easily applied, with an acceptable time complexity; but they still rely on heuristics to certain extent for solving the semantic problem. On the other hand, tree based techniques can avoid the semantic problem with the help of tree structure, for Web page authors tend to cluster texts of different topics into different sub trees. However, general matching on trees is harder than on sequences, hence various restrictive assumptions were made in order to get an acceptable solution.

In this paper, we propose to combine the advantages of tag sequence based techniques and tree based techniques, so as to come up with a hybrid, effective and efficient solution. In particular, our contributions are as follows:

1. We propose an extended sequence representation of Web page, named as *TSReC*(Tag Sequence with Region Code), which can reserve necessary tree structure information. Building from one pass scanning of HTML and region code encoding, it is suitable for both tag sequence based and tree based extraction.
2. We propose an effective algorithm based on *TSReC* for automated Web news content extraction. It contains two procedures, namely, Sequence Matching and Tree Matching. The former one can detect and remove the identical parts of Web news pages, such as navigation bars, copyright notes. The latter one can match and remove the similar structures of Web news pages, such as advertisements and activities. Consequently, our algorithm can differentiate Web news content from others.
3. Empirical evaluation of our algorithm is performed on news pages crawled from real Web sites. The results show that our algorithm is both effective and efficient.

The rest of this paper is organized as follows. In section 2, we review some key Web extraction techniques from the literature, as well as the major work related to Web news extraction. In section 3 the problem of Web new content extraction is defined, and *TSReC* is discussed in section 4 along with an algorithm for building it. In section 5, we discuss how the Web news content extraction can be applied to a search engine, and actually provide the details of our algorithms. Finally, empirical evaluation is studied in section 6, and the conclusion is given in section 7.

2 Related Work

The great demand on Web data extraction has attracted many researchers [16] in recent years, and great efforts have been paid in finding automatic solutions to free people from manual work. Earlier research works were more on semi-automatic tools [10], whereas later ones on automated extraction techniques have become more popu-

lar. However, depending on how to model the HTML Web pages, these techniques mainly fall into two categories: tag sequence based v.s. tree based techniques.

Representational research works on tag sequence based are Stalker [6] and Road-Runner [2]. In Stalker (which also has a commercial version called FETCH), a wrapper program for extraction is learnt from the tag sequence of a Web page. For example, a template based on the fragment "513 Pico, Venice, Phone 1-800-555-1515" may be derived as "513 Pico, *, Phone 1-*-555-1515", which contains two fields for location and district number. By providing enough positive and negative examples, a simple fail-and-release strategy is proposed, which starts from the rigid template, and relaxes the restrictions whenever of the accuracy is found to be not good. As an advantage, utilizing tag sequence enables the authors to apply existing sequence matching techniques. In RoadRunner system, an algorithm is proposed to infer union-free regular expressions that represent page templates, based on tag sequence as well. Regular expression techniques are adopted thus a solid theory foundation exists. However, problems arise when iterating the sequence from one section into another (e.g., from "navigation" to "news body"), due to that there is no extra information in the tag sequence for differentiating them easily. The algorithm in RoadRunner has to try every possible template for the best result, which incurs an unacceptable huge searching space, and therefore leads to an exponential time complexity. In [3], Arvind and Hector proposed an improved version called ExLag with a polynomial time complexity by employing several heuristics. There are other works based on tag sequence-like data structures [7, 8], but still encounter similar problems. Overall, viewing a Web page in HTML as a tag sequence (single words and tags) makes the pattern detection to be easier, but causes handling nesting structures to be harder.

More recently, tree based extraction methods are proposed, such as [1] and [4]. Both of them limit the matching process to work only under sub trees, which helps to differentiate unrelated sections of a Web page. Although sub tree structure brings out some light, pattern reduction turns out to be the dark side. Based on the results of restricted tree edit distance computation process, the work in [1] attaches wildcards to tree nodes and employs heuristics when there is a need to generalize. In [4], an improved version, based on a novel technique named partial tree alignment, is proposed. It can align corresponding data fields in data records without doing wildcards generalizations. However, its assumption on no complex nesting structures limits its use in applications [5], and in special cases adaptation or new techniques have to be made.

Additionally, there are some other related works to ours, which try to utilize the visual cues of Web page to do extraction [9, 11, 12, 13, 14]. Visual cues of a Web page can be derived after a parsing and rendering process. According to visual features, such as layout, font size and color, extraction of specified pieces of Web page content is possible [14]. Further solutions for more complex extraction tasks can also be obtained [9, 11]. However, the feature selection nature of these techniques requires many thresholds or heuristics, which should be trained first and is usually domain specific. An example work in this category is [12], which attempts to automatically extract Web page titles in this category. Comparing to it, our method does not rely on as many heuristics or thresholds.

3 Web News Content Extraction – the Problem

Fig. 1. Example Web news Page from Yahoo, with Navigation Area (Area 1), Events (Area 2), Relate Links (Area 3), and Advertisements (Area 4)

As an example, **Fig. 1** shows a Web news page crawled from Yahoo News, in which we can see that, besides news content, there are other components like navigation areas, events, related links, and advertisements. The objective of Web news content extraction is to find out just *the content*, which is the main part of the Web page after removing the components mentioned above.

One reasonable assumption is that related Web news pages coming from the same Web site share an almost similar page layout and structure, which is usually called the *template*. Actually these web pages are generated by filling a web page template with values queried from the backend database. Therefore most Web data extraction works rely on comparing or matching Web pages that follow the same template. Our method is also based on this idea.

Retrieving Web pages following the same template is possible and practical. Research on clustering or classifying Web pages [1, 15] enables us to do such retrieval in theory, and link analysis techniques in nowadays search engines make it practical. In the Web news domain, as we find out, even simply using the most similar links to the Web pages could achieve acceptable results.

Before we proceed to discussing the algorithms for identifying and extracting the content of Web news, we first give out the definition of *template*.

Definition (Template): A *template* is an incomplete Web page based on which a complete Web page can be generated by filling reserved fields with values. It usually consists of a *common part*, *regular part* and *content part:*

1. *Common part* refers to reserved texts which can not be changed.
2. *Regular part* refers to reserved rigid structure which contains unfilled fields for future values.
3. *Content part* is the reserved area which can be filled with arbitrary html fragments.

Referring back to **Fig. 1**, we have a feel for various parts in the template acccording to the above definition,. Navigation areas (marked by number 1) are the *common parts*, since they are fixed. Events (marked by number 2), relate links (marked by number 3), and advertisements (marked by number 4) are the *regular parts*, because they adhere to the same structure across different Web news pages even though the inner fields may change. The *content parts* are the rest of that page (not marked), which can be freely filled in during generation, and tend to have no fixed patterns.

With the understanding of the *template*, we can divide the Web news content extraction problem into two sub problems, namely, matching for the *common part* and matching for the *regular part*. As we have reviewed in section 2, the former one can be easily solved by sequence matching, whereas the latter one should be better done by tree matching. In order to combine them into the same framework, we have to first design a data structure that is suitable for both techniques.

4   *TSReC*: Tag Sequence with Region Code

In this section we introduce an extended version of tag sequence, namely, Tag Sequence with Region Code (*TSReC*), which is designed for applying both tag-sequence based and tree based extraction algorithms.

As we know, a tag sequence is suitable for extraction but it does not hold any structural information. This backward prevents the utilization of sub trees' information from solving cross trees' ambiguity. For example, by matching on the sequence, we are not able to differentiate the content and advertisement in HTML; but with the help of structural information, the boundaries of them are clear because they are usually resident under different sub trees.

Therefore the basic idea is to extend a tag sequence with extra structural information. In recent database research, the region code in XML processing [17] has proven to be an ideal way to attach structural information in element based storage. With extra storage of a few numbers (region code), all structural relationships can be reserved, such as parent-child, ascent-decedent, and sibling relationships. For our work, we adopt the idea behind the region code, and define *TSReC* as follows.

Definition (TSReC): *TSReC* is a sequence of elements, each of which is defined as
$$TS = <N, RC_b, RC_e, RC_p, RC_l, C>$$
where
1. N is the name of TS, which usually is the same as the HTML tag creating it.
2. RC_b, RC_e, RC_p, and RC_l are region codes, which correspond to begin, end, parent, and level, respectively.

3. *C* is the content of *TS*, which may contain inner HTML tags and text, or be empty.

In **Fig. 2**, there is an example illustration of *TSReC*. At the top part there is a fragment of HTML which shows some links of related articles in science, and there are two categories. In the bottom is its corresponding *TSReC* fragment. With respect to the definition of TSReC, we can see that each element represents a tag in HTML with a corresponding (identical) name. For each of them, four numbers, say RC_b, RC_e, RC_p, and RC_l, are stored in order to keep the structural information. As we will learn in the later algorithm for generating *TSReC*, RC_b and RC_e are begin-end region code identical to XML processing. We use the parent's begin code as the child's RC_p, and also calculate the child's RC_l according to the parent's level. Some TS in *TSReC* may have content consisting of simple HTML tags and texts, which is the same as the conventional tag sequence. As *TSReC* is an extended tag sequence, the sequence matching is supported naturally. On the other hand, the tree matching method can also be applied. Necessary operations, such as calculation of parent-child relationship (by utilizing the RC_p), calculation of ascent-decedent relationship (by utilizing the RC_b and RC_e), are supported. Taking the line 108, line 110 and line 117 as the example, with simple calculation, we know the line 108 and line 110 are under the same sub tree (for $RC_b(108) < RC_b(110) < RC_e(110) < RC_e(108)$), whereas the line 117 in another sub tree (for $(RC_b(117), RC_e(117))$ is not in $(RC_b(108), RC_e(108))$).

Fig. 2. An fragment HTML and its *TSReC* representation

TSReC can be easily built up by one-pass scanning of a Web page. **Fig. 3** shows the algorithm for building it, which is modified from the conventional one for building tag trees. Basically the algorithm scans the Web page tag by tag (line 05, where text is also treat as a tag), and *TSReC* elements are created in lines 06-29 while computing the begin-end region code. According to the different types of tokens (open tag in lines 06-18, close tag in lines 26-29), different actions are taken. Note that a

new TS is created only when a tag comes to break the text flow (line 08, line 14), such as "P", "DIV", "TABLE" and so on. This heuristic helps us get rid of the effect of HTML decoration tags (such as "B", "FONT", "A"), which extraction algorithms usually do not care. Finally, after one pass scanning, a *TSReC* instance is returned as the result.

```
Algorithm buildTSReC(w) /* w as input is a web page */
01 TSReC tsrec /* variable for TSReC holder */
02 TS temp_TS /* temp variable of TS */
03 Stack S /* a stack facility used in the algorithm */
04 int count, level, parent
05 while t = readNextTerm(w) do
06    if t is open Tag then
07         ts = getTop(S)
08         if t breaks text flow then
09              if ts is null then
10                   count = 0, level = 0, parent = 0
11              else
12                   count++, level = ts.RC_b, parent = ts.RC_1+1
13              end if
14              temp_TS = createNewTS(t, count, level, parent)
15              push(S, temp_TS)
16         else // t does not break text flow
17              appendToContent(ts, t)
18         end if
19    elseif t is close Tag then
20         if t breaks text flow then
21              ts = pop(S)
22              count++
23              ts.ie = count
24              append(tsrec, ts)
25         end if
26    else if t is Text then
27         ts = getTop(S)
28         appendToContent(ts, t)
29    end if
30 end while
31 return tsrec
```

Fig. 3. buildTSReC - Algorithm for building TSReC from Web Page

5   Automated Web News Content Extraction

Having defined *TSReC*, in this section we discuss a hybrid method for automated Web news content extraction. We lable our algorithm as hybrid because it combines sequence matching techniques with tree matching techniques, in which the former is used for identifying and removing the *common part*, and the latter is for *regular part*. The overall flow chart of our method is as given in **Fig. 4**.
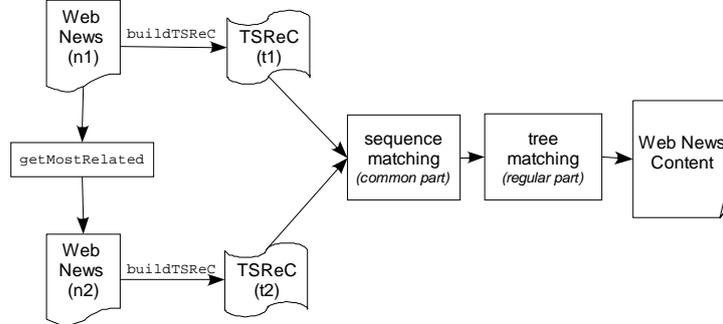
Fig. 4. The Processing Flow of Our Hybrid Method for Web News Content Extraction

The process starts with a Web news page (n1) as its input, and returns the content as its output. Firstly, a function called `getMostRelated` is invoked to get another Web news page probably sharing the same template (cf. section 3). In our evaluation, we simply use the Web page from the related links with its URL most similar to n1's. Then we build *TSReC*s for both Web news pages (i.e., (n1 and n2). After the *TSReC* structures are built up, sequence matching is performed which is followed by tree matching. Each of them will identify and classify specific parts by assigning class marks to elements of *TSReC*. Finally elements with no class marks are output, which are just the news content we need.

5.1 Sequence Matching for *Common Part*

Referring back to the definition in section 3, the target of sequence matching is to find out the *common part* which is supposed identical in two individual Web news pages (cf. Area 1 in **Fig. 1**). Intuitively, we may think that simply comparing two sequences (*TSReC*) can lead us to find out the *common part*. However, things are often a bit more complex, since there are usually more than one *common part*, between each two of which are variable *regular* and *content parts*. The matching process thus becomes not so easy as we have to skip some tags in the sequence to get the best matching result. Due to this reason, we are motivated to adopt a conventional string edit distance calculation algorithm to achieve the goal.

String edit distance calculation is to find out how similar two strings are. A solution based on conventional dynamic programming has a by-product showing the mappings. For example, after the calculation, besides the edit distance between S1("abc") and S2("ac"), we can also know that "*a* in S1 maps to *a* in S2", "*c* in S1 maps to *c* in S2", and "*b* in S1 has no mapping". Taking the advantage of that, we propose an algorithm in **Fig. 5**, based on the dynamic programming solution of string edit distance calculation for sequence matching.

```
Algorithm sequenceMatch(t1, t2)
01 int t1size = sizeof(t1)
02 int t2size = sizeof(t2)
03 int M[t1size+1][t2size+1]
04 M[i][0] = i, i=0,1,2,...,t1size+1
05 M[0][i] = i, i=0,1,2,...,t2size+1
06 for i=1 to t1size do
07    for j=1 to t2size do
08        ts1 = the ith ts of t1
09        ts2 = the jth ts of t2
10        int match = 1
11        if ts1 and ts2 have same tag name and content text then
12            match = 0
13        end if
14        M[i][j] = Min(M[i-1][j-1]+match, M[i-1][j], M[i][j-1])
15        if M[i][j] == M[i-1][j-1]+match then
16            mark matching of ts1 and ts2
17        end if
18    end for
19 end for
```

Fig. 5. Sequence Matching Algorithm on *TSReC* for Common Part

Our sequence matching algorithm takes two *TSReC*s (t1, t2) as the input, does matching for the *common parts* and marks them. Being the same as conventional calculation of string edit distance, our algorithm also uses dynamic programming techniques (lines 03-19). Different from comparing characters in string, our algorithm compares TSs in *TSReC* (line 11). If two TSs have the same tag name and content text, we regard them as they matched, in which case corresponding marking operations are performed (line 16). Otherwise, we move on to look for further matched tags. Note that, in our implementation, we always let t1 be the shorter sequence so as to optimize the algorithm further (ie., let the shorter sequence lead the other loop).

TSReC₁   TSReC₂

```
1: <div>(1, 26, 0, 1) :                    <---> 1: <div>(1, 26, 0, 1) :
2: <form>(2, 25, 1, 2) : <input><table><tr>  <---> 2: <form>(2, 25, 1, 2) : <input><table><tr>
3: <td>(3, 6, 2, 3) : <font><a><font>Yahoo! <a><  <---> 3: <td>(3, 6, 2, 3) : <font><a><font>Yahoo! <a><
4: <div>(4, 5, 3, 4) : <script><script><script><scrip  <---> 4: <div>(4, 5, 3, 4) : <script><script><script><scrip
5: <td>(7, 8, 2, 3) : <spacer>            <---> 5: <td>(7, 8, 2, 3) : <spacer>
6: <td>(9, 10, 2, 3) : <img>             <---> 6: <td>(9, 10, 2, 3) : <img>
7: <td>(11, 12, 2, 3) : <font><input><input><tr>  <---> 7: <td>(11, 12, 2, 3) : <font><input><input><tr>
8: <td>(13, 14, 2, 3) : <a><img>         <---> 8: <td>(13, 14, 2, 3) : <a><img>
9: <td>(15, 18, 2, 3) : <font><font><strong><a>Sign I  <---> 9: <td>(15, 18, 2, 3) : <font><font><strong><a>Sign I
10: <br>(16, 17, 15, 4) : New User?<a>Sign Up    <---> 10: <br>(16, 17, 15, 4) : New User?<a>Sign Up
                        ⋮                                           ⋮
47: <h5>(88, 89, 87, 6) : Secondary Navigation   <---> 47: <h5>(88, 89, 87, 6) : Secondary Navigation
48: <ul>(90, 103, 87, 6) :               <---> 48: <ul>(90, 103, 87, 6) :
49: <li>(91, 92, 90, 7) : <a>Weather News  <---> 49: <li>(91, 92, 90, 7) : <a>Weather News
50: <li>(93, 94, 90, 7) : <a>Space & Astronomy  <---> 50: <li>(93, 94, 90, 7) : <a>Space & Astronomy
51: <li>(95, 96, 90, 7) : <a>Animals & Pets  <---> 51: <li>(95, 96, 90, 7) : <a>Animals & Pets
52: <li>(97, 98, 90, 7) : <a>Dinosaurs & Fossils  <---> 52: <li>(97, 98, 90, 7) : <a>Dinosaurs & Fossils
53: <li>(99, 100, 90, 7) : <a>Biotechnology  <---> 53: <li>(99, 100, 90, 7) : <a>Biotechnology
54: <li>(101, 102, 90, 7) : <a>Energy    <---> 54: <li>(101, 102, 90, 7) : <a>Energy
```

Fig. 6. Sequence Matching Result on *TSReC* – An Example

Fig. 6 shows an example result of sequence matching on two *TSReC*s, which are built up from two related Yahoo News pages. We can see that the *common parts* in each of them have been matched, therefore as the next step we only have to identify and remove the *regular parts*.

5.2 Tree Matching for *Regular Part*

However, differentiating the *regular part* from the *content part* of a Web news page is not as easy as finding *common parts*. Sometimes the mission is even impossible without utilizing semantics, if the *regular parts* in the template are too flexible to fill anything. So it is reasonable to assume that the *regular part* has rigid format structure while the *content part* has not. For example, the *regular part* is usually like "<UL>(<LI><A>(text)</A>)*</UL>", whereas the *content part* can be any free HTML fragment. Based on our observation, this assumption is common in real-life Web pages for news, and therefore our algorithm takes advantage of it in differentiating the regular part.
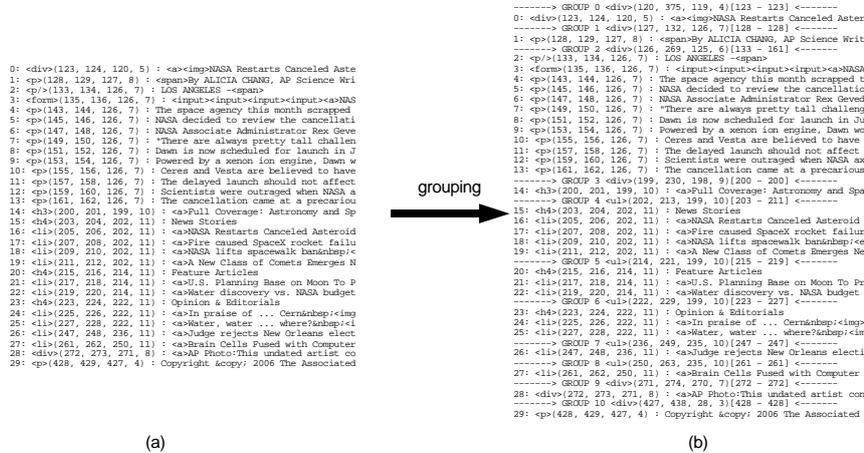
```
0: <div>(123, 124, 120, 5) : <a><img>NASA Restarts Canceled Aste
1: <p>(128, 129, 127, 8) : <span>By ALICIA CHANG, AP Science Wri
2: <p/>(133, 134, 126, 7) : LOS ANGELES -<span>
3: <form>(135, 136, 126, 7) : <input><input><input><input><a>NAS
4: <p>(143, 144, 126, 7) : The space agency this month scrapped
5: <p>(145, 146, 126, 7) : NASA decided to review the cancellati
6: <p>(147, 148, 126, 7) : NASA Associate Administrator Rex Geve
7: <p>(149, 150, 126, 7) : "There are always pretty tall challen
8: <p>(151, 152, 126, 7) : Dawn is now scheduled for launch in J
9: <p>(153, 154, 126, 7) : Powered by a xenon ion engine, Dawn w
10: <p>(155, 156, 126, 7) : Ceres and Vesta are believed to have
11: <p>(157, 158, 126, 7) : The delayed launch should not affect
12: <p>(159, 160, 126, 7) : Scientists were outraged when NASA a
13: <p>(161, 162, 126, 7) : The cancellation came at a precariou
14: <h3>(200, 201, 199, 10) : <a>Full Coverage: Astronomy and Sp
15: <h4>(203, 204, 202, 11) : News Stories
16: <li>(205, 206, 202, 11) : <a>NASA Restarts Canceled Asteroid
17: <li>(207, 208, 202, 11) : <a>Fire caused SpaceX rocket failu
18: <li>(209, 210, 202, 11) : <a>NASA lifts spacewalk ban <
19: <li>(211, 212, 202, 11) : <a>A New Class of Comets Emerges N
20: <h4>(215, 216, 214, 11) : Feature Articles
21: <li>(217, 218, 214, 11) : <a>U.S. Planning Base on Moon To P
22: <li>(219, 220, 214, 11) : <a>Water discovery vs. NASA budget
23: <h4>(223, 224, 222, 11) : Opinion & Editorials
24: <li>(225, 226, 222, 11) : <a>In praise of ... Cern <img
25: <li>(227, 228, 222, 11) : <a>Water, water ... where? <i
26: <li>(247, 248, 236, 11) : <a>Judge rejects New Orleans elect
27: <li>(261, 262, 250, 11) : <a>Brain Cells Fused with Computer
28: <div>(272, 273, 271, 8) : <a>AP Photo:This undated artist co
29: <p>(428, 429, 427, 4) : Copyright &copy; 2006 The Associated
```

grouping →

```
-------> GROUP 0 <div>(120, 375, 119, 4)[123 - 123] <-------
0: <div>(123, 124, 120, 5) : <a><img>NASA Restarts Canceled Aster
-------> GROUP 1 <div>(127, 132, 126, 7)[128 - 128] <-------
1: <p>(128, 129, 127, 8) : <span>By ALICIA CHANG, AP Science Writ
-------> GROUP 2 <div>(126, 269, 125, 6)[133 - 161] <-------
2: <p/>(133, 134, 126, 7) : LOS ANGELES -<span>
3: <form>(135, 136, 126, 7) : <input><input><input><input><a>NASA
4: <p>(143, 144, 126, 7) : The space agency this month scrapped t
5: <p>(145, 146, 126, 7) : NASA decided to review the cancellatio
6: <p>(147, 148, 126, 7) : NASA Associate Administrator Rex Geved
7: <p>(149, 150, 126, 7) : "There are always pretty tall challeng
8: <p>(151, 152, 126, 7) : Dawn is now scheduled for launch in Ju
9: <p>(153, 154, 126, 7) : Powered by a xenon ion engine, Dawn wo
10: <p>(155, 156, 126, 7) : Ceres and Vesta are believed to have
11: <p>(157, 158, 126, 7) : The delayed launch should not affect
12: <p>(159, 160, 126, 7) : Scientists were outraged when NASA ax
13: <p>(161, 162, 126, 7) : The cancellation came at a precarious
-------> GROUP 3 <div>(199, 230, 198, 9)[200 - 200] <-------
14: <h3>(200, 201, 199, 10) : <a>Full Coverage: Astronomy and Spa
-------> GROUP 4 <ul>(202, 213, 199, 10)[203 - 211] <-------
15: <h4>(203, 204, 202, 11) : News Stories
16: <li>(205, 206, 202, 11) : <a>NASA Restarts Canceled Asteroid
17: <li>(207, 208, 202, 11) : <a>Fire caused SpaceX rocket failur
18: <li>(209, 210, 202, 11) : <a>NASA lifts spacewalk ban <e
19: <li>(211, 212, 202, 11) : <a>A New Class of Comets Emerges Ne
-------> GROUP 5 <div>(214, 221, 199, 10)[215 - 219] <-------
20: <h4>(215, 216, 214, 11) : Feature Articles
21: <li>(217, 218, 214, 11) : <a>U.S. Planning Base on Moon To Pr
22: <li>(219, 220, 214, 11) : <a>Water discovery vs. NASA budget
-------> GROUP 6 <ul>(222, 229, 199, 10)[223 - 227] <-------
23: <h4>(223, 224, 222, 11) : Opinion & Editorials
24: <li>(225, 226, 222, 11) : <a>In praise of ... Cern <img>
25: <li>(227, 228, 222, 11) : <a>Water, water ... where? <im
-------> GROUP 7 <ul>(236, 249, 235, 10)[247 - 247] <-------
26: <li>(247, 248, 236, 11) : <a>Judge rejects New Orleans electi
-------> GROUP 8 <ul>(250, 263, 235, 10)[261 - 261] <-------
27: <li>(261, 262, 250, 11) : <a>Brain Cells Fused with Computer
-------> GROUP 9 <div>(271, 274, 270, 7)[272 - 272] <-------
28: <div>(272, 273, 271, 8) : <a>AP Photo:This undated artist con
-------> GROUP 10 <div>(427, 438, 28, 3)[428 - 428] <-------
29: <p>(428, 429, 427, 4) : Copyright &copy; 2006 The Associated
```

(a)   (b)

Fig. 7. (a) Different Parts after Sequence Matching. (b) Grouping Results on Different Parts

In our approach, before doing tree matching, we do grouping first. **Fig. 7** gives out an illustration on how grouping is performed. **Fig. 7**(a) is the various parts of a specific Web page derived after the sequence matching; as we can see, tags are organized one by one, being not aware of structures. The grouping process tries to find out which of them are under the same sub trees. As shown in **Fig. 7**(b), for example, line 13 and line 14 are in different groups, which were not known in **Fig. 7**(a). This grouping process is necessary in order to do tree matching subsequently; its algorithm is given in **Fig. 8**.

The grouping algorithm takes a list containing different parts (ie., the rest of the *TSReC* excluding *common parts*) as its input, and returns groups as the output. Each group is a TS (called group parent) with two numbers (called group region), namely, group beginning and group ending. Any TS whose region codes (beginning and ending) are in the group region belongs to that group. The grouping method is to simply check the parent and tree level (lines 06-07) of siblings. If siblings share the same parent and tree level, they are under the same sub tree so we add them into the same group (line 08) by simply extending the group region. Otherwise, a new group will be created (lines 10-11).

```
Algorithm subTreeGroup(dp)
01 List groups /* list for holding groups */
02 /* each group is a list of TS,
03    and initial a group using first element of dp */
04 List group = createGroup(dp[0], dp[0].level, dp[0].parent)
05 for i=1 to sizeof(dp) do
06    if dp[i].level == group.level &&
07       dp[i].parent = group.parent then
08          append(group, dp[i])
09    else
10          append(groups, group)
11          group = createGroup(dp[i], dp[i].level, dp[i].parent)
12    end if
13 end while
14 return groups
```

Fig. 8. Grouping Algorithm

After grouping, we get sub trees which correspond possibly to the *regular* or *content parts*. So we have to differentiate the *regular part* from the others. As we have discussed, the determination of whether a sub tree in a Web page is a *regular part*, is measured by weather there is a sub tree in the other Web page sharing the same rigid pattern. Accordingly, we have the tree matching algorithm as shown in **Fig. 9**.

```
Algorithm treeMatch(t1, t2, rsm)
01 List<TS> dp1 = getDifferentPart(t1, rsm)
02 List<TS> dp2 = getDifferentPart(t2, rsm)
03 List<TS> groups1 = getSubTreeGroup(dp1)
04 List<TS> groups2 = getSubTreeGroup(dp2)
05 int nextMatch = 0
06 for i=0 to sizeof(groups1) do
07    group1 = compactGroup(getGroup(groups1, i))
08    for j=nextMatch to sizeof(groups2) do
09       group2 = compactGroup(getGroup(groups2, j))
10       if groupMatch(group1, group2) then
11             markNotContent(group1)
12             nextMatch = j+1
13             break
14       end if
15    end for
16 end for
17 return groups1
```

Fig. 9. Tree Matching Algorithm on TSReC for Regular Part

Basically the tree matching algorithm tries to find for each group (sub tree) a matched group in the other Web page, which is done in a nest-loop (lines 06-16). The outer loop iterates on t1 (the Web page to be extracted), and the inner loop iterates on t2 (the Web page as reference). The parameter rsm is the sequence matching result, which is necessary in extracting different parts (lines 01-02). However, we can do a simple optimization according to the following observation: Usually *regular parts* share the same order even in different Web pages (e.g. "event" appearing before "advertisement" in one Web page seldom appears in reverse order in another Web

page). This optimization in our algorithm is as reflected in line 08 (ie, to start matching after the previous matching position).

A notable function in the tree matching algorithm is the one named `compactGroup` (line 07). It is designed to handle repeatable fields in the *regular part*. For example, referring back to the example of in **Fig. 2**, where "Science News" is a *regular part* which usually contains several lines for titles and links of news in identical topics. However, it is a repeatable field, which can have proper instances according to the underlying database. Since we target to find rigid patterns, multiple instances of a repeatable field should be reduced, and return in the form of "<UL>(<LI><A>(text)</A>)*</UL>". In the algorithm of `compactGroup`, we thus only check whether siblings share the same sequence patterns. **Fig. 10** shows the `compactGroup` algorithm which is of a fairly simple process.

After the tree matching process, we have identified both the *common part* (by sequence matching) and the *regular part*. Thus the rest of the Web page is unmarked and is the content part needed. By simply returning this part, we get the Web news content as desired.

```
Algorithm compactGroup(groups)
01 int i=1, j
02 while i<sizeof(groups) do
03      j = i+1
04      while j<sizeof(groups) do
05        if patternMatch(groups[i], groups[j]) then
06            j++
07          else break
08        end if
10      end while
11      i++
12 end while
```

Fig. 10. Algorithm of Compacting Group

6 Empirical Evaluation

As part of the proposed approach, we have built a prototype system for Web news content extraction, upon which empirical evaluation has been conducted. In this section, we describe the implementation of the prototype system, testing data bed used, evaluation method and evaluation results, and explanations of the results.

The Prototype System: All the proposed algorithms in this paper, as well as the *TSReC* data structure of our Web news content extraction system, are implemented in Java with the help of HTML Parser [18]. The prototype system accepts two Web news content pages (one to be extracted, one as the reference) sharing the same template, and returns the HTML fragment of Web news content. As we discussed in previous section, finding two Web news pages of the same template is not hard, and in fact, we just take the one having the most similar URL to the reference page's URL. We reserve the output in HTML form, which can be directly delivered to indexing facilities for parsing and indexing.

Testing Data Bed: The testing data bed used in our evaluation is manually crawled from real-life Web news sites. We have collected news pages from up to 50 Web news sites, covering news of politics, business, sports, entertainment, and life. For each Web news site, we randomly collect one page, then running a script to get from the related links the Web news page sharing probably the same template with the source. The 50 Web news sites, which are 25 in English and 25 in Simplified Chinese, are selected from the online categories of famous search engines (google.com and baidu.com). The names of the Web sites are given in **Table 1**.

Evaluation Method: As an empirical evaluation, our evaluation is conducted in the following steps. We first extract Web pages from each Web site by running our crawler. Then we manually check whether the result is the content of Web news. We take the content part as the content of Web news according to the definition in section 3, while the semantics of text is not considered.

Evaluation Results and Explanation: The result of evaluation is as shown in **Table 1**. For each Web site, we list out the URL[ψ], as well as two evaluation results (R1 and R2). R1 and R2 can be of the value S or F: S means that the extraction is successful and the content is correctly extracted (as judged by manual checking); F means that the extraction fails, which may be caused by various reasons.

In particular, F1 means that the extraction failed with no output. The reason is that the *content part* is incorrectly identified as the regular part, for its content may be simply a sequence of text sentence (no highlighted sub section titles, no hyperlinks for key words). So the `compactGroup` function takes a wrong action on it. This kind of situation is somehow common as there are 5 cases in our data set. However, it is easy to be handled by using a set of simple heuristic rules, such as by judging how long the text is (eg., when it exceeds specific threshold, we stop matching it). Actually, those simple heuristic rules are actually applied, as shown by the later results (R2).

F2 means that the extraction failed because the HTML source of the Web news page contains some invisible text, thus affecting the matching process. These texts are usually for Dynamic HTML effects, such as popup menu. By using a simple heuristic to remove it from the original source, the problem is easily fixed.

However, there are still error types F3 and F4 for which currently we do not have a solution. Specifically, F3 means that there are regular parts not in rigid forms, and F4 means that even the regular parts are highly dynamic. These errors do not follow the assumption we made before, therefore corrupt the extraction process. Fortunately, these errors are not common, and they are left as a future work.

Overall, as we can see from **Table 1**, the accuracy of our method is generally high (19/25=76% and 21/25=84% before applying the simple heuristic rules, and after is 23/25=92% and 24/25=96%). Given that our method does not rely on any a threshold nor machine learning knowledge, it is practically feasible and suitable to be applied in real life search engines.

[ψ] Due to the space limitation, however, the URLs are only indicative in Table 1. Readers are referred to [19] for the detailed URLs.

Table 1. The Result of Evaluation On Real-life Web News Sites

	Chinese Web News			English Web News		
	URL	R1	R2	URL	R1	R2
1	news.sina.com.cn...	S	S	news.yahoo.com...	S	S
2	beijing.qianlong...	F_3	F_3	news.yahoo.com...	S	S
3	news.qq.com...	S	S	www.cbc.ca...	S	S
4	politics.people....	F_2	S	www.cbc.ca...	F_3	F_3
5	news.tom.com...	S	S	www.cnn.com...	S	S
6	news.xinhuanet.c...	S	S	www.cnn.com...	S	S
7	www.gmw.cn...	S	S	www.msnbc.msn.co...	S	S
8	news.163.com...	S	S	www.msnbc.msn.co...	S	S
9	news.espnstar.co...	S	S	today.reuters.co...	S	S
10	www.southcn.com...	F_1	S	today.reuters.co...	F_1	S
11	news.21cn.com...	S	S	www.un.org...	S	S
12	heilongjiang.nor...	S	S	www.un.org...	S	S
13	www.cnhan.com...	F_1	S	www.cbsnews.com...	S	S
14	gb.chinabroadcas...	S	S	www.cbsnews.com...	S	S
15	www.yzdsb.com.cn...	S	S	articles.news.ao...	S	S
16	sports.sohu.com...	S	S	articles.news.ao...	S	S
17	news.sports.cn...	F_2	S	www.usatoday.com...	F_1	S
18	www.chinanews.co...	F_4	F_4	www.usatoday.com...	S	S
19	economy.enorth.c...	S	S	today.reuters.co...	S	S
20	news.17173.com...	S	S	today.reuters.co...	S	S
21	www.daynews.com....	S	S	www.latimes.com...	S	S
22	sports.nen.com.c...	S	S	www.latimes.com...	S	S
23	www.lanews.com.c...	S	S	www.heraldsun.ne...	F_1	S
24	news.huash.com...	S	S	www.heraldsun.ne...	S	S
25	www.jhnews.com.c...	S	S	smh.com.au...	S	S
A		19/25	23/25		21/25	24/25

7 Conclusion

In this paper we have studied the Web news content extraction problem and proposed an automated extraction algorithm for it. Our algorithm exhibits a hybrid method applying both sequence matching and tree matching techniques. Built on top of *TSReC* – a variant of tag sequence previously proposed by ourselves, the hybrid method benefits from combining the advantages of both sequence matching and tree matching. Empirical evaluation conducted shows that our method is highly effective and efficient. Because of the automatic nature of our method, it should be fairly straightforward for us to integrate it into a real life search engine, such as Yahoo! Or Google, as a preprocessing procedure to improve indexing quality. In addition, we plan to further improve the algorithm and enhance our prototype system to make it more robust, able to handle more types of errors as mentioned in section 6.

References

1. Reis, D. Golgher, P., Silva, A., Laender, A. Automatic Web news extraction using tree edit distance, WWW-04, 2004.
2. Crescenzi, V., Mecca, G. and Merialdo, P. RoadRunner: Towards automatic data extraction from large web sites. VLDB-01, 2001.
3. Arasu, A. and Garcia-Molina, H. Extracting Structured Data from Web Pages. SIGMOD-03, 2003.
4. Zhai, Y., and Liu, B. Web data extraction based on partial tree alignment. WWW-05, 2005.
5. Bing Liu and Yanhong Zhai. NET - A System for Extracting Web Data from Flat and Nested Data Records., WISE-05, 2005
6. Muslea, I., Minton, S. and Knoblock, C.. A hierarchical approach to wrapper induction.. Agents-99, 1999.
7. Wang, J., and Lochovsky, F. Data extraction and label assignment for Web databases. WWW-03, 2003.
8. Chang, C. and Lui, S-L. IEPAD: Information extraction based on pattern discovery. WWW-10, 2001.
9. Zhao, H., Meng, W., Wu, Z., Raghavan, V. and Yu, C. Fully automatic wrapper generation for search engines.. WWW-05, 2005.
10. Alberto H. F. Laender, Berthier A. Ribeiro-Neto, Altigran Soares da Silva, and Juliana S. Teixeira. A brief survey of web data extraction tools. SIGMOD Record, 31(2):84–93, 2002.
11. Bing Liu, Robert Grossman, Yanhong Zhai. Mining Data Records in Web Pages. KDD-2003, 2003
12. Yunhua H., Guomao X. , Ruihua S., Guoping H., Shuming S., Yunbo C., and Hang L., Title Extraction from Bodies of HTML Documents and its Application to Web Page Retrieval, SIGIR2005, 2005
13. D. Cai, S. Yu, J.-R. Wen, and W.-Y. Ma. Vips: a vision-based page segmentation algorithm. Technical Report MSR-TR-2003-79, Microsoft, 2003.
14. Can Lin, Zhang Qian, Xiaofeng Meng, and Wenyin Liu. Postal address detection from web documents. In WIRI, pages 40–45, 2005.
15. Crescenzi, V., Mecca, G. and Merialdo, Wrapping-Oriented Classification of Web Pages. SAC2002, 2002
16. Bing Liu, WISE-2005 Tutorial: Web Content Mining. WISE2005, 2005
17. Q.Li, B.Moon. Indexing and Querying XML Data for Regular Path Expressions. VLDB 2001, 2001
18. Dhaval Udani, HTMLParser project, available at http://sourceforge.net/projects/htmlparser.
19. Yu Li, Evaluation of Hybrid Extraction Method, available at http://idke.ruc.edu.cn/hybrid.