Eliminating Noisy Information from News Websites
and Extraction of the News article

by
Shekhar Palta
IMIT IV Edition
Scuola Superiore Sant Anna
Pisa, Italy

Tutor In University : Dr. Tommaso Cucinotta

Tutor in the Company : Dr. Antonio Gulli, Mr. Antonio Savona, Mr. Giuseppe Ottaviano, Mr. Luca Foschini
ABSTRACT

A typical web page containing a news article contains many information blocks. Apart from the main content blocks, it usually has blocks such as navigation panels, copyright and privacy notices, and advertisements (for business purposes and for easy user access). We call these blocks that are not the main content blocks of the page the noisy blocks. In this paper, we propose an algorithm for the effective extraction of the text of the news article out of the news web page. We have devised this algorithm based on the following observation: In a given news Web site, noisy blocks usually share some common content, while the main content blocks of the pages are often diverse in their actual contents. Based on this observation, we propose a tree structure similar to the one proposed in [2] but with much less storage of data and faster computation which we call as a Page Content Tree (PCT) for a single web page and a Site Content Tree (SCT) for a family of news web pages from the same news site. An SCT can be built by combining the PCTs of various news pages on the news site in a way which will be described later in this paper. We then describe an information-based measure of computing the “relevance” of a node in the SCT to determine whether it belongs to the desired content or the noisy content. At the end of the paper, we will show the results and experimental evaluations which show the high accuracy of our algorithm.

General Terms
Algorithm, Design, Experimentation, Theory.

Keywords
Noise Detection, Noise Elimination, Data Extraction.

1. INTRODUCTION

The rapid expansion of the Internet has made the WWW a popular place for searching and reading news. Data mining on the Web for news sites thus becomes an important task for discovering useful knowledge or information from the Web. However, useful information on the news websites is often accompanied by a large amount of noise such as banner advertisements, navigation bars, copyright notices, etc. Although such information items are functionally useful for human viewers and necessary for the news web site owners, they often hamper automated news article extraction and Web data mining, e.g., Web page clustering, classification, information retrieval and information extraction. Web noises can be grouped into two categories according to their granularities:

- Global noises: These are noises on the Web with large granularity, which are usually no smaller than individual pages. Global noises include mirror sites, legal/illegal duplicated Web pages, old versioned Web pages to be deleted, etc.

- Local (intra-page) noises: These are noisy regions/items within a Web page. Local noises are usually incoherent with the main contents of the Web page. Such noises include banner advertisements, navigational guides, decoration pictures, etc.

In this paper we present a domain-oriented approach to web data extraction and discuss its application to automatically extracting news from various news web sites we have on the web. Indeed there are a plethora of sites providing daily updated news and there is a growing need for tools that can keep track and extract news articles from the news sites in a completely automatic manner. In this work, we focus on detecting and eliminating local noises in news web pages to improve the performance of Web mining for news web pages, e.g., Web page clustering and classification. This work is motivated by a practical application. During my stage work, I was asked to design an application which would extract only the news article out of a number of web pages which would be then useful in searching for keywords that a user enters to search for news on the world wide web.

Despite the importance of extraction of news article from news pages, little work has been done keeping News Sites in mind. The news web pages are considerably different from other web sites and subsequently, the information extraction algorithms should also be different and tuned to the news web sites. The difference can be illustrated by the fact that in most non-news sites the noisy information is mostly in the form of advertisements, navigation bars, footers but in news sites, in addition to all these the noisy information is also represented as links and snippets of related news items whose text is closely related to the news article content which should not be identified as an important part of the page.

As shown in Figure 1, a CNN news page has been partitioned into various blocks. We can see a navigation bar at the left, a block at the top carrying the CNN logo, then we have a form just under the top block featuring the “search news” option. Our algorithm will basically assign a relevancy value to each node in the DOM tree and that value along with the link to word ratio of a node determines whether a node is carrying a news article or not.

Our contributions:

- A new tree structure, called Page Content Tree, is proposed to capture the actual contents of the news Web pages in a news site. An information (or entropy) based measure is also introduced to evaluate the importance of each element node in the Site Content Tree (SCT), which helps us to eliminate noises in the news page.

- Experimental results show that the proposed page cleaning tree indicate noises and which parts of the content tree contain the technique is able to improve the results of Web data mining for news pages dramatically. It also outperforms the existing template based cleaning technique given in [4] by a large margin.
also domain knowledge to generate classification rules. However, these techniques are not automatic. They require a large set of manually labeled training data and are within the scope of BODY. Each The cleaning method in [4] is not optional or disjunction elements. A modification to a tag path causes the wrapper to easily break. Therefore, manually coding and maintaining the wrapper can be all-consuming work. A tailored wrapper can extract data from a particular web site because the targeted web pages are usually generated from the same template, and therefore share identical navigation bars.

In [4], Web page cleaning is defined as a frequent template detection problem. They propose a frequency based data mining algorithm to detect templates and views those templates as noises. since all the viewable parts are within the scope of BODY. Each The cleaning method in [4] is not concerned with the context of a Web site, which can give useful clues for page cleaning.

Moreover, in [4], the partitioning of a Web page is prefixed by considering the number of hyperlinks that an HTML element has. This partitioning method is simple and useful for a set of Web pages from different Web sites, while it is not suitable for Web pages that are all from the same Web site because a Web site typically has its own common layouts or presentation styles, which can be exploited to partition Web pages and to detect noises.

[8][11] propose some learning mechanisms to recognize banner ads, redundant and irrelevant links of Web pages. However, these techniques are not automatic. They require a large set of manually labeled training data and also domain knowledge to generate classification rules.
Many methods proposed for the template problem relies on the concept of page segmentation. Since it is usually simple to visually separate template content from informative content because they both occur in well-defined, non-overlapping area, (they are not scattered one into another) a lot of work have been devoted to attempt to segment a web page into smaller but more cohesive pagelets. The notion of pagelet does provide a new angle for the granularity of information retrieval. However, it is difficult to obtain the optimum threshold value for determining whether a sub parse tree is indeed a pagelet. For instance, Xiaoli Li et al. [16] proposed segmenting web page into topically more focused MIU (Micro Information Unit). Their algorithm employs text font property and term set similarity, building HTML page tag tree by matching heading with text, and adjacent similar paragraphs as MIU. Bar-Yossif et al. [4] employ a notion of pagelet to segment a web page too. In their case, a pagelet is determined by the number of hyperlinks in a HTML element and web page cleaning is defined as a frequent template detection problem. They propose a frequency based data mining algorithm to detect templates and views those templates as noises. The cleaning method in [4] is not concerned with the context of a web site, which can give useful clues for page cleaning. Moreover, in [4], the partitioning of a web page is prefixed by considering the number of hyperlinks that an HTML element has. This partitioning method is simple and useful for a set of web on a human-labeled test set. They highlight the applications of template detection by showing that removing templates as a preprocessing step boosts the accuracy of standard web mining tasks on their datasets, by as much as 140% on duplicate detection, and 18% on web page classification. Further, they show that the gains obtained by using their page-level template detection approach are substantially greater than those obtained by using the more expensive site-level approach.

3. THE PROPOSED TECHNIQUE

The proposed cleaning technique is based on the analysis of both the layouts and the actual contents (i.e., texts, images, etc.) of the Web pages in a given Web site. Thus, our first task is to find a suitable data structure to represent the actual contents of the Web pages in the site. We propose a Content Tree (CT) for this purpose. Below, we start by giving an overview of the DOM (Document Object Model) tree, which is commonly used for representing the structure of a single Web page, and showing that it is insufficient for our purpose. We then present the content tree, which is followed by our entropy calculations.

3.1 DOM tree

Each HTML page corresponds to a DOM tree where tags are internal nodes and the detailed texts, images or hyperlinks are the leaf nodes. Figure 2 shows a segment of HTML codes and its corresponding DOM tree. In the DOM tree, each solid rectangle is a tag node. Notice that our study of HTLM Web pages begins from the BODY tag since all the viewable parts are within the scope of BODY. Each node is also attached with its display properties. For convenience of analysis, we add a virtual root node without any attribute as the parent tag node of BODY in the DOM tree.

Although a DOM tree is sufficient for representing the layout or presentation style of a single HTML page, it is hard to study the overall presentation style and content of a set of HTML pages and to clean them based on individual DOM trees. Thus, DOM trees are not enough in our cleaning work. We need a more powerful structure for this purpose. This structure is critical because it needs to represent each node of a DOM tree and we also associate a lot of attributes to each node whose values are calculated as explained in the later sections. The values of the attributes ultimately decides the fact whether a node is a part of a template or not. We introduce the concept of a Content Tree for this purpose.

The construction of the Content Tree is shown in Figure 3. Node $d_1$ and $d_2$ are two DOM trees of different pages of the same site. They are merged to produce a merged tree as shown. The Content Tree thus obtained keeps track of the number of pages a particular node / set of nodes occurs over a set of pages. The process followed to build a Site Content Tree from a collection of pages from the same news web site is listed below.

![Figure 2. An example of a DOM tree (all tags not shown)](image)

![Figure 3 : A Content Tree built with combination of two DOM Trees](image)
A comprehensive list of nodes is listed in Table 1 which were pruned out from the DOM trees of each news web page. Also, in each of the web pages we replaced occurrences of &amp;, &lt; etc with spaces.

### 3.3 Content Tree (CT)

We now define a content tree which consists of content nodes and candidate nodes.

**Definition**: A Content Node represents a layout, which has two components, denoted by $(C_c,n)$, where $C_c$ is a sequence of candidate nodes (see below), and $n$ is the number of pages that has this particular set of nodes at this node level.

In Figure 3, the content node (in a dashed line rectangle) $P$-$IMG$-$P$-$A$ consists of four candidate nodes $P$, $IMG$, $P$, $A$.

**Definition**: A candidate node $E$ has three components, denoted by $(Tag, C_c)$ where:

- $Tag$ is the tag name, e.g., `TABLE` and `IMG`;
- $C_c$ is a set of content nodes below $E$.

Note that a candidate node corresponds to a tag node in the DOM tree, but points to a set of child content nodes $C_c$ (see Figure 3). For convenience, we usually denote a candidate node by its tag name, and a content node by its sequence of tag names corresponding to its candidate node sequence.

### 4. Determining Noisy Elements in the Content Tree (CT)

We will define a node as "noisy" or irrelevant if it occurs a high number of times across the set of news pages from a web site. We use a set of mathematical formulas to compute the relevance of each of the nodes. The value depends on the number of times that particular node occurs across the set of web pages we have from the same site. The higher the importance, the more likely of that node being a node carrying the important part i.e the news article.

In the example of Figure 4, the shaded parts of the SCT are more likely to be noises since their occurrences are highly regular and fixed and hence less important. The double lined Table element node has many child content nodes, which indicate that the candidate node is likely to be important. That is, the double-lined Table is more likely to contain the main contents or the news content of the pages. Specially, the double-lined Text element node is also meaningful since its content is diverse although its presentation style is fixed. Let the SCT be the tree built using all the pages of a news Web site.

![Figure 4: An example of a Site Content tree (SCT)](image-url)
where \( E_j \) is a candidate node in \( S_i.C_c \), and \( k = |S_i.C_c| \), which is the number of candidate nodes in \( S_i \).

In (2), \( \gamma \) is the attenuating factor, which is set to 0.9. It increases the weight of \( \text{NodeRel}(E) \) when \( l \) is large. It decreases the weight of \( \text{NodeRel}(E) \) when \( l \) is small. This means that the more child content nodes an element node has, the more its composite relevance is focused on itself, and the fewer child content nodes it has, the more its composite relevance is focused on its descendents. Leaf nodes are different from internal nodes since they only have actual content with no tags. We define the composite relevance of a leaf element node based on the information in its actual text.

**Definition**: Here we define a method to compute the composite relevance of a leaf node. A leaf node is defined as a node which contains any kind of text. We follow a completely different procedure for a leaf node because a leaf node, unlike any other nodes in the DOM tree, does not contain any child nodes so the formula relating to Composite relevance will not be applicable to a leaf node.

For a leaf element node \( E \) in the SCT, let \( l \) be the number of words appearing in \( E \) and let \( m \) be the number of pages containing \( E \), the composite relevance of \( E \) is defined by:

\[
\text{CompRel}(E) = \begin{cases} 
\frac{1}{\sum_{i=1}^{j} H(a_i)} & \text{if } m = 1 \\
1 - \frac{1}{l} & \text{if } m > 1 
\end{cases}
\]

where \( a_i \) is an actual feature of the content in \( E \), where \( E \) is the leaf node. \( H(a) \) is the information entropy of \( a \) within the context of \( E \),

\[
H(a_i) = -\sum_{j=1}^{m} p_{ij} \log_m p_{ij}
\]

where \( p_{ij} \) is the probability that \( a_i \) appears in \( E \) of page \( j \). Note that if \( m = 1 \), it means that only one page contains \( E \), then \( E \) is a very important node, and its \( \text{CompRel} \) is 1 (all the values of \( \text{CompRel} \) are normalized to between 0 and 1).

Calculating composite importance for all element nodes and style nodes can be easily done by traversing the SST. We will not discuss it further here due to brevity concerns.

**Input**: Root Element \( R \) of the SCT computed

**Input**: Root Element \( Rp \) of the incoming news page

**MapResult( R, Rp)**:

1. if \( R \) is noisy :
2. return 
3. end if
4. for children in \( R.children \):
5. if \( R.children == Rp.children \):
6. if \( R \) is listed important :
7. return content or leaf node immediately under \( Rp \)
8. \( r \) belongs to \( R.children \)
9. \( rp \) belongs to \( Rp.children \)
10. for each pair \( (r, rp) \):
11. MapResult(\( r, rp \))
12. end for
13. else if \( R.children != Rp.children \):
14. if \( R \) is listed important :
15. returnContent(\( Rp \))
16. end if
17. end if
18. end for

**Algorithm 1**: To return the content of the news article from an incoming webpage

**4. Method Of Noise Detection**

We have just explained the formation of the Site Content Tree (SCT) and subsequently the computation of the values of Composite Relevance (see section 3) for each node. We record all these values for each node in the special tree data structure, which we have designed to store the characteristics and the relation of each node to the rest of the nodes. We skip the exact description of the data structure used by us here.

The method followed by us to extract the news content coming from a news page is described in Algorithm 1. Basically, we keep traversing the DOM tree of the new page and consecutively we keep going down the SCT also and keep matching the children of each node in the DOM tree as well as in the SCT so as to track the path to be taken down to the leaf node.

As we go down the SCT we keep checking the composite relevance of the node we are at and if the composite importance of the node in SCT is above a pre-set threshold and if the Link-to-word ratio (see section 4.1) is also below a given threshold we return the content directly under the node we are at in the DOM tree of the new page. The selection of a node in the SCT as important depends primarily on two factors in our algorithm:

1. The composite relevance of the node in the SCT should be higher than a given threshold. After a number of experiments, we set a threshold of 0.24. A value of composite relevance below 0.24 means that the corresponding node in the DOM tree is one which is not a node carrying the news article. It may be one of a navigation bar or of the various types of extra information that may be available on the web page.

2. The Link-word ratio of the node must be below a specified threshold. We have used a threshold of 0.2 for the Link-Word ratio (see section 4.1 for details on the computation of the Link-Word ratio).
5. The Overall Algorithm

We list the overall algorithm in Algorithm 2 below.

1. Randomly crawl n pages from a news web site and store it in a database
2. Set SCT as NULL with a virtual root R
3. for each page P in the n pages do :
   4. MergeIntoSCT(P)
   5. end for
4. R is the root of the SCT
5. CalcNodeRelevance ( R )
6. CalcNodeCompRelevance ( R )
7. for each target web page do :
   8. MapResult (R,Rp)
   9. end for
6. end for

Algorithm 2 : The overall algorithm

6. Experimental Evaluation

Using the robust interface of Django 0.96.2 , we designed a web interface wherein we displayed the dirty web page (original web page ) in a frame and also displayed the extracted or the clean text after applying our cleaning algorithm next to the former. This gave us a perfect platform to visually track the progress of our algorithm. We could easily track how our algorithm was performing in extracting data by visually comparing the results.

6.1 Our Database

We constructed a comprehensive database of about 230 news web sites with about 30 to 70 web pages crawled from each of those 230 news web sites giving us a comprehensive database to work and evaluate our algorithm's performance.

We used a diverse database comprised of equal number of feeds from four categories i.e. sports, politics, business and health.

I used the comprehensive news feeds provided from the database that is used for BigNews (www.news.ask.com)

6.2 Evaluation with Digramic Bayesian Text Classifier

To evaluate the performance of our algorithm we made use of the widely respected industry standard Digramic Bayesian Text Classifier. It is available as a python module by the name of dbacl.

We followed the following strategy :

1. training / testing of dirty web pages (original web pages)
2. training / testing of cleaned web pages after extraction of news text from dirty web pages

We used 10 original (dirty) web pages of each category to train the bayesian classifier for each of the four categories of news. We also used 10 cleaned and extracted text from each category of news to train classifier for each category of cleaned pages.

The results of the classification are shown in Table 2 and Table 3 and it clearly shows that the experimental evaluation for cleaned text outperforms the accuracy for classification of uncleaned news web page by a huge margin.

6.3 Future Work

The future work of this project involves a lot of improvements that can be done to improve the accuracy of the extracted text.

1. The extracted text right now also includes the name of the author and the presence of dates and the likewise text that occurs as part of the news article and it becomes very difficult for the algorithm to prune out these phrases from the extracted text. So, a solution we have thought of is to use the concept of a suffix array to detect the regular and repeated occurrence of certain unimportant words over the set of extracted news article from the same web site. This has already been implemented by a colleague at Ask and we are working to integrate it with the present algorithm.
<table>
<thead>
<tr>
<th></th>
<th>TOTAL</th>
<th>CORRECT</th>
<th>WRONG</th>
<th>ACCURACY</th>
</tr>
</thead>
<tbody>
<tr>
<td>POLITICS</td>
<td>50</td>
<td>32</td>
<td>18</td>
<td>64 %</td>
</tr>
<tr>
<td>BUSINESS</td>
<td>50</td>
<td>36</td>
<td>14</td>
<td>72%</td>
</tr>
<tr>
<td>HEALTH</td>
<td>50</td>
<td>40</td>
<td>10</td>
<td>80%</td>
</tr>
<tr>
<td>SPORTS</td>
<td>50</td>
<td>30</td>
<td>20</td>
<td>60%</td>
</tr>
</tbody>
</table>

Table 2: Results of Classification for Dirty (original) web pages

<table>
<thead>
<tr>
<th></th>
<th>TOTAL</th>
<th>CORRECT</th>
<th>WRONG</th>
<th>ACCURACY</th>
</tr>
</thead>
<tbody>
<tr>
<td>POLITICS</td>
<td>42</td>
<td>40</td>
<td>2</td>
<td>95.20%</td>
</tr>
<tr>
<td>BUSINESS</td>
<td>51</td>
<td>50</td>
<td>1</td>
<td>98.04%</td>
</tr>
<tr>
<td>HEALTH</td>
<td>50</td>
<td>49</td>
<td>1</td>
<td>98%</td>
</tr>
<tr>
<td>SPORTS</td>
<td>50</td>
<td>50</td>
<td>0</td>
<td>100%</td>
</tr>
</tbody>
</table>

Table 3: Results of Classification for Cleaned or Extracted Text with much higher accuracy

2. We intend to use this particular algorithm to extract news content for the entire set of news web sites on the world wide web for a variety of languages and to bring it to use for BigNews (www.news.ask.com).

7. Acknowledgements

I would really like to thank the support and help that has been offered to me both by the tutors at Scuola Superiore Sant Anna and at Ask.com, IAC Search and Media during the stage work. I would specially like to thank Giuseppe Ottaviano and Luca Foschini from ASK who never failed to answer any of my persistent doubts and were always patient with a novice like me in the field of reverse templating, during the entire course of my stage work. I would also like to acknowledge the kind support rendered to me by Dr. Antonio Gulli and Mr. Antonio Savona at ASK.

I would also like to thank Dr. Giuseppe Lipari, Head of IMIT at Scuola Superiore Sant Anna for his unconditional presence for all the Master students.

I would like to include a special mention and would like to thank Dr Claudio Manfroni, the Secretary Of our course at Scuola Superiore Sant Anna who never ceased to help me in any of the issues I faced during and before the stage.

8. References


