Can We Learn a Template-Independent Wrapper for News Article Extraction from a Single Training Site?*

Junfeng Wang[1] Chun Chen[1]† Can Wang[1] Jian Pei[2]
Jiajun Bu[1] Ziyu Guan[1] Wei Vivian Zhang[3]
[1]Zhejiang Key Lab. of Service Robot, College of Computer Science, Zhejiang University, China.
[2]School of Computer Science, Simon Fraser University, Canada.
[3]Microsoft Research, Redmond, WA.
[1]{wangjunfeng, chenc, wcan, bjj, guanzh}@zju.edu.cn
[2]jpei@cs.sfu.ca [3]wzha@microsoft.com

ABSTRACT

Automatic news extraction from news pages is important in many Web applications such as news aggregation. However, the existing news extraction methods based on template-level wrapper induction have three serious limitations. First, the existing methods cannot correctly extract pages belonging to an unseen template. Second, it is costly to maintain up-to-date wrappers for a large amount of news websites, because any change of a template may invalidate the corresponding wrapper. Last, the existing methods can merely extract unformatted plain texts, and thus are not user friendly. In this paper, we tackle the problem of template-independent Web news extraction in a user-friendly way. We formalize Web news extraction as a machine learning problem and learn a template-independent wrapper using a very small number of labeled news pages from a single site. Novel features dedicated to news titles and bodies are developed. Correlations between news titles and news bodies are exploited. Our template-independent wrapper can extract news pages from different sites regardless of templates. Moreover, our approach can extract not only texts, but also images and animates within the news bodies and the extracted news articles are in the same visual style as in the original pages. In our experiments, a wrapper learned from 40 pages from a single news site achieved an accuracy of 98.1% on 3,973 news pages from 12 news sites.

Categories and Subject Descriptors

H.4.m [**Information Systems Applications**]: Miscellaneous—*Data extraction, Web*

General Terms

Algorithms, Experimentation.

Keywords

Data extraction, Web mining, classification.

1. INTRODUCTION

Reading news is a popular behavior of Internet users according to the surveys by the Pew Internet & American Life Project. The November 2005 survey[1] shows that 46% of Internet users are accustomed to reading Web news daily. Energized by this huge number of users, some new Web services emerged to automatically extract news articles from thousands of online news portals. Among them, Web news aggregation systems like Google News, together with some automatically generated news list pages, are gradually winning users from traditional online news portals. Furthermore, most of the news Websites nowadays are not designed for disabled people. Unrelated information on a Web page like navigational links and advertisements are keeping disabled people out of bounds, particularly those with a visual impairment who access Web news via screen readers. It would significantly improve the accessibility of Web news if the extracted news instead of raw HTML is fed to screen readers.

However, extracting news articles from thousands of online news portals is a very challenging task. This is not only because the Web is highly heterogeneous, but also because there are no rigid guidelines on the publication of online news [16]. The existence of various noises on news pages, such as advertisements, makes accurate news extraction very difficult. Furthermore, while news articles are embedded in semi-structured HTML, poor quality HTML pages may hinder the development of robust extraction tools. For example, Zhao *et al.* [24] found that, due to the loose HTML grammar, the majority of Web pages are not well formed. The Opera Software, one of the browser manufacturers, reported that only 4.13% of 3,509,180 Web pages in 3,011,668 domains completely comply with the HTML standard[2]. Like browsers, HTML-based Web information extraction methods should be extremely robust to all anomalies in HTML

*This work was supported by National Key Technology R&D Program (2008BAH26B02 & 2007BAH11B06) and Key S&T Projects of Zhejiang Province (2007C13019).

†Corresponding author

KDD'09, June 28–July 1, 2009, Paris, France.

[1]http://www.pewinternet.org/~/media/Files/Reports/2005/PIP_SearchData_1105.pdf.pdf
[2]http://dev.opera.com/articles/view/mama-key-findings/

pages [24]. Unfortunately, building a highly error tolerant HTML parser is far from trivial.

As most Web news pages nowadays are generated from some underlying structured sources, it is intuitive to assume the existence of some template structures in news pages from a specific news portal. Some previous template-level approaches like Tree Edit Distance (TED) exploit structured similarities in HTML pages and try to generate wrappers [16]. However, the generated wrappers can only work properly for news pages belonging to the previously seen templates. Any subtle changes in the underlying HTML structures are likely to invalidate the wrappers. Furthermore, the previous template-based wrappers such as TED can only extract plain texts from news pages. While this may meet the needs of news extraction in early days of the Internet, it is quickly outdated by the rapid development of network infrastructure, which has helped the World Wide Web evolve into a powerful means of disseminating multimedia information on a global scale [10]. The observation is further verified by Shen *et al.* [19], who suggested that images were increasingly being embedded in Web pages and the semantics of these images were closely related to the surrounding text. This suggests that various content resources like images embedded in news articles should not be excluded from the extraction since they may convey important meanings and are likely to be closely related to the news content.

To eliminate the template dependency in DOM wrappers, Zheng *et al.* [25] proposed a template-independent news extraction technique by exploiting the visual features of online news pages. In their approach, the news title and body in a news page are identified as visual blocks and extracted accordingly. This approach is insensitive to the underlying structural changes as long as the visual perception of the news page remains consistent. However, it may still fail to maintain the integrity of an extracted news article as it does not treat a news article as an inseparable unit. By considering only the leaf nodes of a visual block tree as the candidates for extraction, this approach is likely to leave out the news title or some fragments of the news body. In [25], block-level evaluation is used to measure the accuracy of news extraction, by which the extraction of a page with a misidentified title and a fragmented news body will still be rated towards a high accuracy if most of the blocks in the page are correctly extracted. Furthermore, both the extraction of news titles and that of bodies use the same features and model. As a result, zero or more than one title may be extracted from a single news page. For applications like news aggregation systems, identifying and extracting the unique title from a news page is highly important.

Contrary to template-based wrappers, human beings can always accurately identify the news article from a news page. This is because the news pages are usually designed to adapt to people's reading habits. Moreover, the fact that human can easily differentiate news titles from news bodies suggests that news titles and news bodies come with different features [25]. Note that, there is also correlation between the news title and the news body in a news page. For example, the news title and the news body are often very close in terms of vertical distance, and often overlap largely in horizontal direction. If these features of news pages can be fully exploited, it is possible to learn a template-independent wrapper using only news pages from a single news portal so that it can accurately extract news articles from various news portals.

Figure 1: T_{TITLE} on a news page.

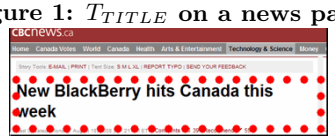


Figure 2: T_{BODY} on a news page.

In this paper, we tackle the problem of template independent Web news extraction in a user-friendly way, and make the following contributions. First, we develop a template independent wrapper that is able to accurately extract news articles from various news portals on the Web. Once the wrapper is learned using only a very small number of pages from a single news site, we are able to extract news articles from news pages in various sites. Second, two models are separately trained to extract news titles and bodies, respectively. Moreover, the correlation between the news title and the body in a news page is exploited to achieve more accurate title extraction. This is a significant improvement over [16] where the title extraction is the major difficulty. Third, the problem of Web news extraction is extended by incorporating rich Web contents. Specifically, we extract not only plain texts of the news as the current approaches do, but also images and animates within the news bodies. Furthermore, the extracted news articles will remain in the same visual styles as their specialized design in the original pages (that is, with the same color, size, etc.).

2. PROBLEM STATEMENT AND FRAMEWORK

Let P denote a news page, T_{DOM} denote the DOM tree built from P, and T denote a subtree of T_{DOM}. T_{TITLE} denotes the news title subtree (referred to as title subtree hereafter) as boxed off in Figure 1. T_{BODY} denotes the news body subtree (referred to as body subtree hereafter) as boxed off in Figure 2.

Formally, a subtree T is T_{TITLE} if the news title is contained in T but is not contained in any child subtree of T. A subtree T is T_{BODY} if the whole news body is contained in T but is not contained in any child subtree of T.

The news extraction problem is that, given any news page P, identify and extract T_{TITLE} and T_{BODY} from T_{DOM}, respectively.

The framework of our approach is shown in Figure 3. It consists of three components.

Figure 3: Overview of our approach.

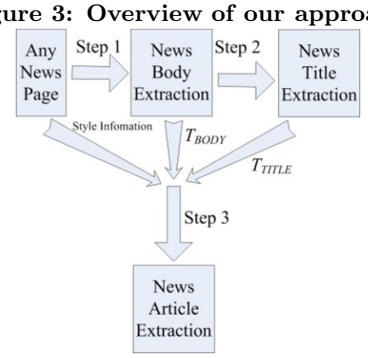


Figure 4: Sample noise: story highlights.

Figure 5: Example of noise: subtitle.

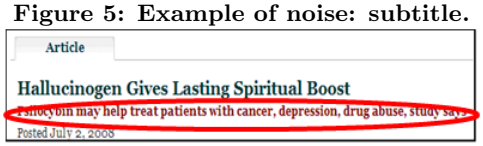
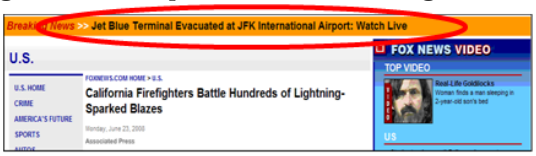


Figure 6: Example of noise: breaking news bar.



News Body Extraction: Using both spatial features and content features dedicated to T_{BODY}, we learn a body subtree identification model. A body subtree extraction algorithm is proposed to extract T_{BODY} from any given news page P by using the learned model.

News Title Extraction: extracting news titles is not as easy as it appears to be since there are various types of noises around news titles, such as, the story highlights in Figure 4, the subtitle and the author name in Figure 5, and the breaking news bar in Figure 6. An example of heavy noise is shown in Figure 9.

Particularly, the <title> element in HTML is used to indicate the title of a Web page. However, the texts in the <title> element here are far different from the news title. The <title> element is not mandatory and may be absent in some pages as reported by Hu *et al.* [7]. Even if the <title> element appears, it is often far different from the news title. Figure 7 shows the news title of a CBS news page. The corresponding HTML code of the <title> element in this page is shown in Figure 8. It is obvious that the <title> element contains many noises besides the news title.

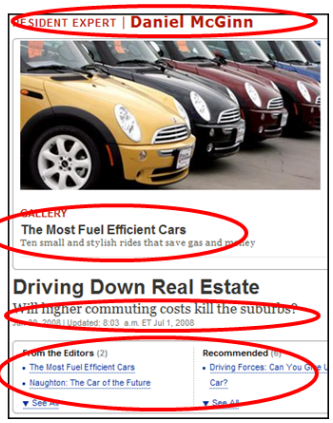
In the news title extraction step, using spatial and content features dedicated to T_{TITLE}, we learn a title subtree identification model. A title subtree extraction algorithm is proposed to extract T_{TITLE} from any given P by using the learned model.

Figure 7: A news title.

Figure 8: HTML code of a <title> element.

```
<title>
Who Really Is Sarah Palin?, CBS Evening News: The Mother Of
Five Now Catapulted To The National Stage - CBS News
</title>
```

Figure 9: Various noises around a news title.



News Article Generation: DOM tree remains highly editable and can be easily reconstructed back into a complete Web page [5]. Although the visual style of a DOM element can be specified in various ways in HTML code, it has to be rendered to the user by the browser rendering engine after all. The final visual style rendered can be extracted with the help of the Web browser. After extracting T_{TITLE} and T_{BODY} from T_{DOM}, we combine these two subtrees with the visual style information to construct a new Web page presenting the extracted news article.

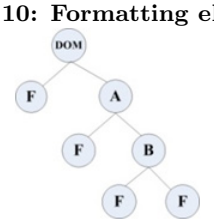
3. NEWS BODY EXTRACTION

In this section, we describe the features dedicated to body subtrees, and then introduce a learning model to extract body subtrees.

3.1 Feature Representation of Body Subtrees

News bodies are mostly occupied by contiguous text paragraphs. This suggests that the content features are important in identifying T_{BODY}. However, in a short page, user comments may contain more text than T_{BODY}. Even in a long page, when discussing hot topics, users may write comments longer than T_{BODY}. The dominance of comments in text length also gives spammers and advertisers good incentives to spam comments as observed by Jindal *et al.* [9]. Spammers can publish very long contents as they wish, for example, repeat their advertisements many times. To this end, we adopt spatial features as well. For example, if a subtree T at the bottom of a page contains many texts, T is probably not the news body but instead the user comments.

Figure 10: Formatting element.



We use both content features and spatial features to represent body subtrees.

3.1.1 Content Features

Texts in news pages are usually carefully arranged in a format convenient for reading via formatting elements. Formatting elements are HTML elements mainly used for formatting texts, including the paragraph element (<p>), the bold font element (<b>), the new line element (
), the italic font element (<i>), and the highlight element ().

Let FEA_i denote the i-th formatting element in T_{DOM}. For a subtree T, let R denote the root element of T, and FE denote the collection of formatting elements which are present in the child elements of R. The following two features related to formatting elements, *FormattingElementsNum* and *FormattedContentLen*, are used to describe the content of T in M_{BODY}.

FormattingElementsNum denotes the size of FE. This feature is normalized by the total number of formatting elements in T_{DOM}. Taking Figure 10 as an example, F represents a formatting element, while A and B are not formatting elements. For the subtree rooted at A, the *FormattingElementsNum* is 1 and will be normalized by 4.

FormattedContentLen is the total length of texts contained in FE. It is normalized by $\sum_i FormattedContentLen(FEA_i)$.

3.1.2 Spatial Features

With the help of the browser, the bounding rectangle of T denoted by *Rect* can be obtained. Spatial features of T include the following four features in M_{BODY}.

RectLeft and *RectTop* are the coordinates of the upper left corner of *Rect*. *RectWidth*, *RectHeight* are the width and the height of *Rect*, respectively. Such spatial features are absolute features since they directly use the absolute values. However, using absolute values may make it hard to compare the feature values from different Web pages as discussed by Song *et al.* [20]. By using the width and the height of the whole page to normalize the absolute features, we transform them into relative spatial features as follows.

$$\{RectLeft/PageWidth, RectTop/PageHeight,$$
$$RectWidth/PageWidth, RectHeight/PageHeight\}$$

However, the above normalized spatial features lead to another problem. For some long pages whose heights are much larger than the screen height, after normalization, some important rectangles on the top part may be transformed into rectangles located at the very top of the page with quite small height. In such a case, the spatial features of these important rectangles are very similar to the spatial features of some unimportant rectangles such as advertisements in short pages. Intuitively, for a long news page, the content in the first two screens is the most important. We should not normalize them using the height of the whole page. Width normalization does not suffer from this problem since very few pages are much wider than the popular screen width. Accordingly, the four absolute spatial features are normalized as follows:

$$RectLeft = \frac{RectLeft}{ScreenWidth}, RectTop = \frac{RectTop}{TwoScreensHeight}$$

$$RectWidth = \begin{cases} \frac{RectWidth}{ScreenWidth}, \\ \quad if \quad RectRight < ScreenWidth; \\ \frac{ScreenWidth - RectLeft}{ScreenWidth}, \quad otherwise \end{cases}$$

$$RectHeight = \begin{cases} \frac{RectHeight}{TwoScreensHeight}, \\ if \quad RectBottom < TwoScreensHeight; \\ \frac{TwoScreensHeight - RectTop}{TwoScreensHeight}, otherwise \end{cases}$$

where *ScreenWidth*, *ScreenHeight* and *TwoScreensHeight* are constants.

3.2 Learning to Identify Body Subtrees

3.2.1 A Support Vector Machine Approach

Identifying the news body subtree among a number of subtrees in a page can be regarded as a classification problem. To solve the problem, a body subtree identification model denoted by M_{BODY} is built based on Support Vector Machines (SVM) developed by V. Vapnik [21]. M_{BODY} is a function which maps the feature values of T to the estimated probability of being T_{BODY}. It can be formalized as follows:

$$f :< features \quad of \quad T > \rightarrow \{0, +1\}$$

where T takes value 1 if it is a body subtree, and takes value 0 otherwise.

In the following, we describe how to learn a body subtree identification model. Suppose n labeled examples are collected that belong to two classes 0, +1. Each example is a d-dimensional feature representation of the subtree, and y_i indicates whether $\vec{x_i}$ is a body subtree or not. The training set can be described as follows:

$$\{(\vec{x_1}, y_1), (\vec{x_2}, y_2), ..., (\vec{x_n}, y_n)\},$$
$$\vec{x_i} \in \mathbb{R}^d, y_i \in \{0, +1\}, i = 1, 2, ..., n.$$

We aim to learn a function to discriminate the body subtrees and the non-body subtrees while the generalization error can be minimized. V. Vapnik [21] showed that the function with this property is the one having the maximum margin between the two classes. Given the training examples, the SVM classifier builds a decision function as follows:

$$y_p(\vec{x}) = sign(\sum_{i=1}^{n} \alpha_i y_i K(\vec{x_i}, \vec{x}) + b)$$

where K is the kernel function. The output of the decision function is assigned to $\vec{x}$ as its label. A remarkable property of this equation is that often only a subset of the points will be associated with non-zero α_i. These points are called the support vectors which are the closest points to the separating hyperplane and lie at the same distance from either side of the hyperplane.

Now the class label of a subtree can be predicted by the decision function. However, there is one critical drawbacks in such a straightforward application of SVM. In a news page, there may be zero, one, or multiple subtrees classified as the news body. In the news body learning problem, we want to extract exactly one news body subtree.

To overcome the problem, we convert the function value to posterior probability . This posterior probability, combined with rules described in the next subsection, will be used to identify body subtrees. Several ways of approximation of the posterior probabilities for SVM are proposed in the literature. Particularly, Platt [15] approximated the posterior probability as follows:

$$P(y = +1|\vec{x}) = \frac{1}{1 + exp(Ay'_p + B)},$$

$$y'_p(\vec{x}) = \sum_{i=1}^{n} \alpha_i y_i K(\vec{x_i}, \vec{x}) + b$$

where A and B are two parameters obtained during the training process.

3.3 Identifying News Body Subtrees

Now we introduce our algorithm for extracting T_{BODY} from T_{DOM} as shown in Figure 11. We aim to find the subtree with maximum probability estimate of being T_{BODY}. T is the current subtree to traverse, and M_{BODY} is the learned body subtree identification model. The algorithm has two steps: 1) Generate T_{BODY} candidates. 2) Apply M_{BODY} on every candidate and select the one with the highest posterior probability. We use the following observations.

Observation 1. The top border of news bodies are always inside the first screen. A news body is unlikely to be far away from the news title. Readers are very likely to get confused if they cannot find the news body at all in the first screen. As mentioned before, some user comments might be even longer than the news body in a news pages. Many user comments whose top borders are outside the first screen will be excluded for news body candidates. This observation helps improve both the efficiency and the accuracy. If the top border of T is outside the first screen, it is impossible for any of T's child subtrees to be inside the first screen, and thus there is no need for further traversal on T.

Observation 2. The area of news bodies cannot be too small. The function *BigEnough* checks whether the area of T is larger than the predefined minimum value for being news body candidates. If T is not large enough, it is impossible for any of its child subtrees to meet the minimum area requirement. Thus no more traversal on T is needed.

In the second step, for each candidate T, the function *PredictProbability* extracts the feature values of T, and then predicts the probability estimate of being T_{BODY} using M_{BODY}.

4. NEWS TITLE EXTRACTION

This section describes the novel features dedicated to title subtrees, and introduces a learning model to extract them.

4.1 Feature Representation of Title Subtrees

Web page designers usually organize the page layout in a reasonable way to emphasize the news title part. Therefore, M_{TITLE} adopts spatial features such as position and size. On the other hand, the contents of T are also helpful in identifying T_{TITLE}. Therefore, M_{TITLE} also adopts content features.

Figure 11: The Body Subtree Extraction Algorithm.

```
BodySubtreeExtraction(M_BODY, T)
 1  let List ← an empty list
 2  IdentifyBodyCandidates(T, List)
 3  let n ← the number of subtrees in List
 4  let L_i ← the i-th subtree in List
 5  let P_MAX ← 0 and T_BODY ← NIL
 6  for i ← 1 to n
 7     do let P ← PredictProbability(M_BODY, L_i)
 8        if P > P_MAX
 9           then P_MAX ← P
10                T_BODY ← L_i
11  return T_BODY

IdentifyBodyCandidates(List, T)
 1  if not TopInScreen(T) or not BigEnough(T)
 2     then return
 3   add T to List
 4  let n ← the number of child subtrees of T
 5  let T_i ← the i-th child subtree of T
 6  for i ← 1 to n
 7     do IdentifyBodyCandidates(List, T_i)
```

4.1.1 Spatial Features

Spatial features of T include: *RectLeft*, *RectTop*, *RectWidth*, *RectHeight*, *Overlap*, *Distance*, and *Flat* .

RectLeft, *RectTop*, *RectWidth* and *RectHeight* describe the position and the size of T. As discussed in Section 3.1.2, the whole page size is not adopted for normalizing these four spatial features. Although news pages usually span multiple screens, it is a convention of news page design to put news titles inside the first screens of pages. So we use the screen size in the normalization, that is,

$$RectLeft = \frac{RectLeft}{ScreenWidth}, \qquad RectTop = \frac{RectTop}{ScreenHeight}$$

$$RectWidth = \begin{cases} \frac{RectWidth}{ScreenWidth}, \\ \quad if \quad RectRight < ScreenWidth; \\ \frac{ScreenWidth - RectLeft}{ScreenWidth}, \quad otherwise \end{cases}$$

$$RectHeight = \frac{RectHeight}{ScreenHeight}$$

Overlap describes the horizontal overlap between T and T_{BODY}. News titles and bodies are usually close to each other in the horizontal direction. It is seldom the case that a news title is put at the left (right) of the Web page, while the news body is put at the right (left). This feature is normalized by the width of T_{BODY}.

Distance describes the vertical distance between T and T_{BODY}. Similarly, the news title is unlikely to be far away from the news body in vertical distance. This feature is normalized by the predefined parameter *ScreenHeight*.

Flat describes the shape of T. Let min be the shorter edge of the bounding rectangle of T, and max be the longer one. *Flat* is defined to be the ratio min/max. This feature is based on the observation that news titles are always bounded by long narrow rectangles.

4.1.2 Content Features

The following features are used to represent the contents of T: *FontSize*, *EndWithFullStop*, *WordNum*.

Figure 12: The Title Subtree Extraction Algorithm.

```
TitleSubtreeExtraction(M_TITLE, T)
 1  let List ← an empty list
 2  IdentifyTitleCandidates(T, List)
 3  let n ← the number of subtrees in List
 4  let L_i ← the i-th subtree in List
 5  let P_MAX ← 0 and T_TITLE ← NIL
 6  for i ← 1 to n
 7    do let P ← PredictProbability(M_TITLE, L_i)
 8       if P > P_MAX
 9         then P_MAX ← P
10              T_TITLE ← L_i
11  return T_TITLE

IdentifyTitleCandidates(List, T)
1  if not TopInScreen(T)
2    then return
3  if WholeInScreen(T) and NoAnchorText(T)
4     and NotCategoryName(T)
5    then add T to List
6  let n ← the number of child subtrees of T
7  let T_i ← the i-th child subtree of T
8  for i ← 1 to n
9    do IdentifyTitleCandidates(List, T_i)
```

FontSize is the font size of the root element of T. It is based on the convention that news titles are always displayed in some font sizes large enough to be distinguished from news bodies. This feature is normalized by the largest font size in the first screen.

EndWithFullStop describes whether the text contained in T ends by a period. By convention, a period is usually used to denote the end of a sentence, while a title hardly ends up with a period. This feature helps to distinguish news titles from sentences.

WordNum describes the number of words in the text contained in T. This feature exploits the fact that news titles do not contain paragraphs of texts. It is normalized by a predefined parameter *MaxWordNum* which is set to 60 in our experiments.

4.2 Learning to Identify Title Subtrees

Identifying news title subtree among a number of subtrees in a page can be regarded as a classification problem. To solve the problem, a title subtree identification model denoted by M_{TITLE} is built based on nonlinear SVM with Gaussian RBF kernel. M_{TITLE} is a function which maps the features of T to probability estimate of being T_{TITLE}, and can be formalized as follows:

$$f :< features \quad of \quad T >\rightarrow \{0, +1\}$$

The learning process of M_{TITLE} is similar to that of M_{BODY} as discussed in Section 3.2.1.

The algorithm for extracting title subtrees is shown in Figure 12. We aim to find the subtree with the maximum probability estimate of being the title subtree. T is the current subtree to traverse, and M_{TITLE} is the learned title subtree identification model. The algorithm has two steps: 1) Generates T_{TITLE} candidates. 2) Apply M_{TITLE} on every candidate and find the T with the maximum probability estimate. We use the following observations in the first step:

Observation 1. News titles are entirely inside the first screen. Readers are very likely to get lost if they cannot find the news title in the first screen. The function *WholeInScreen* checks whether T is entirely inside the first screen. If not, T cannot even be a candidate for T_{TITLE}.

Observation 2. News titles cannot contain any anchor text. Hyperlinks lead readers to some other different pages. If a news title contains anchor texts, then it increases the probability that a user leaves the current news page by incidental clicks.

Figure 13: Part of a news page in TIME.COM.

Figure 14: Multiple category names on a news page.

Observation 3. In some news Web pages, category names may appear more than once above the news body. A huge number of news pages in news sites are organized into categories. It is common to see the category names above the news title as shown in Figure 13. The category names and the news title are similar in many ways such as font size, position, etc. The font size of the category name "Entertainment" is even larger than that of the news title "Ben Folds". It is a challenging task to distinguish them. Intuitively, if a reader is attracted by the current news page, he may be interested in reading more news in the same category. Thus the page designers should take this into consideration and provide a hyperlink for the readers to jump to the category page. This is a common practice of news page design. Once if the category name appears above the news title, it would probably appear somewhere else again in some navigation bar. For example, the hyperlink for the category "Entertainment" is at the top of the page as shown in Figure 14. This helps to distinguish news titles from category names. *NotCategoryName* checks whether there is a hyperlink above the news body containing the same text with T.

5. EXPERIMENTS

5.1 Experimental Setup

We use a dataset of 4,013 news pages[3] crawled from 12 online news sites as shown in Table 1. We manually compared the extracted news pages with the original news pages to evaluate the performance of our method. The accuracy measure used in all our experiments is defined as follows:

$$Accuracy = \frac{\#\ of\ correctly\ extracted\ pages}{\#\ of\ testing\ pages}$$

Both title subtrees and body subtrees in a small number of pages are manually labeled. The required number of labeled

[3]Both data and program are available at http://eagle.zju.edu.cn/KDD'09/

Figure 15: The news labeling tool.

Table 1: Detail of dataset

Name	Site	Pages
1. CNN	edition.cnn.com	234
2. BBC	news.bbc.co.uk	360
3. GC.CA	news.gc.ca	455
4. YAHOO	news.yahoo.com	318
5. CBC	www.cbc.ca	408
6. CBSNEWS	www.cbsnews.com	291
7. FOXNEWS	www.foxnews.com	292
8. NEWSWEEK	www.newsweek.com	316
9. SKY	www.sky.com	330
10. TIME	www.time.com	219
11. USATODAY	www.usatoday.com	437
12. USNEWS	www.usnews.com	353
Total number		4013

pages varies in different experiments. We developed a tool for labeling the news titles and bodies as shown in Figure 15. Given a news page, the title subtree and the body subtree can be selected on the DOM tree. After labeling, the title subtree is boxed off in red dotted box and body subtree is boxed off in black dashed box.

Our objective is to learn a template-independent wrapper for the entire news domain using a small number of training pages from a single site. The state-of-the-art method proposed by Reis *et al.* [16] is template-dependent, and thus is unable to extract the entire news domain using only a single training site. We use their published results although the dataset is different. In the V-Wrapper approach by Zheng *et al.* [25], a block-level evaluation is used to measure the performance of news extraction, by which a page with misidentified title and fragmented news body extraction will still be rated with high accuracy if most of the blocks are correctly extracted. We evaluate our approach at the page-level. That is, we consider how many pieces of news (both title and body) are correctly extracted whereas Zheng *et al.* [25]considered how many blocks were correctly extracted. Our objective is quite different from theirs.

We conducted the experiments on a PC with a 2.3 GHz AMD Athlon processor and 1,012 MB RAM.

5.2 Wrapper Training Experiment

This experiment is designed to explore the following two important issues on training the wrapper:

1. How does the number of training pages affect extraction accuracy? The number of training pages is closely related to the cost of manual labeling. A method is impractical if it requires a large number of training examples.

Table 2: Testing results of the learned wrappers

Training Site	M10	M20	M40
1. CNN	98.90%	99.40%	98.70%
2. BBC	98.90%	98.90%	99.40%
3. GC.CA	96.20%	97.30%	98.90%
4. YAHOO	99.20%	98.90%	98.30%
5. CBC	98.90%	98.90%	98.90%
6. CBSNEWS	98.70%	98.30%	98.30%
7. FOXNEWS	99.40%	98.70%	98.70%
8. NEWSWEEK	98.50%	97.90%	98.70%
9.SKY	98.70%	99.20%	99.20%
10.TIME	99.60%	98.90%	99.40%
11. USATODAY	98.90%	98.50%	98.30%
12. USNEWS	99.60%	99.60%	99.60%

2. How does the choice of the training site affect the extraction accuracy? This issue is important because we aim to train a template-independent wrapper using pages from a single site.

In each news site, we label 40 pages among which 10, 20, 40 training examples are randomly chosen to train 3 wrappers (W10, W20, W40), respectively. In each site, we randomly choose 40 pages for testing. The 480 testing pages (40 pages × 12 sites) are different from the training pages. Each of the 36 learned wrappers is applied to extract news articles from the 480 testing pages. Finally, 17,280 extracted news articles (36 wrappers × 480 testing pages) are evaluated by a user study.

Table 2 shows the testing results. All the 36 wrappers trained from a small number of pages achieve an accuracy higher than 96%, which demonstrates the stability of our approach. Also, our approach only requires a very small number of training examples to achieve this performance. Although there is no significant difference between using different sites for training, we find that some sites are more suitable for training wrapper. For example, all the three wrappers trained from USNEWS achieve the highest accuracy, which implies that USNEWS is a good training site. Therefore, USNEWS is chosen to be the training site for the remaining experiments.

5.3 Experiments with Different Features

Spatial and content features are used in the identification models of both title subtrees and body subtrees. The goal of this experiment is to explore the effects of following three feature sets on the extraction of news titles and bodies:

- spatial features only (SPATIAL)
- content features only (CONTENT)
- all features (ALL).

As explained in Section 5.2, the trained USNEWS-W40 wrapper is used in the following experiments. For each site, we randomly downloaded 40 Web news pages for testing. Note that, these testing pages are different from the pages used for training. As a result, 1,440 extracted news titles and 1,440 extracted news bodies (40 pages× 12 sites×3 feature sets) are evaluated by a user study. We first evaluate the effects of the above three feature sets on news title extraction. For each feature sets, a wrapper is learned. The testing results are shown in Table 3.

Using spatial features alone performs the worst. This is

Table 3: Feature contribution for title extraction

Feature set	Features	Accuracy
SPATIAL	*RectLeft, RectTop, RectWidth, RectHeight, Overlap, Distance, Flat*	74.80%
CONTENT	*FontSize, EndWithFullStop, WordNum*	81.80%
ALL	*All the above features*	100%

Table 4: Feature contribution for body extraction

Feature set	Features	Accuracy
SPATIAL	*RectLeft,RectTop, RectWidth,RectHeight*	1.90%
CONTENT	*FormattingElementsNum, FormattedContentLen*	99.37%
ALL	*All the above features*	99.79%

probably due to noises such as subtitles. Subtitles are quite similar to the news titles in terms of spatial properties, which make them difficult to be distinguished. When content features are used alone, section titles within the news bodies are possibly identified as news titles. By combining both spatial and content features, the best performance can be achieved.

In the next we evaluate the effects of the three feature sets on news body extraction. Again, for each feature set, a wrapper is learned. Table 4 shows the testing results.

Spatial features perform extremely badly for news body extraction. When spatial features are used alone, the first paragraph is very likely to be extracted as news body. Content features perform very well because most of the news pages have regular structure. However, as discussed in Section 3, in some special cases like photo gallery pages, spatial features may be useful. Therefore, both spatial and content features should be used together to get the best performance.

5.4 Large Scale Experiment

In this experiment, we aim to explore the performance of our template-independent wrapper over thousands of pages. The trained USNEWS-W40 wrapper is used in this experiment as explained in Section 5.2. As shown in Table 1, we collected 4,013 news pages from 12 news sites. 40 pages are used to train the USNEWS-W40 wrapper and the remaining 3,973 pages are used for testing. The 3,973 extracted news articles are evaluated by a user study.

Table 5 shows the testing results. Although the number of testing pages increases from 480 in the wrapper training experiment to 3,973 in this experiment, the wrapper still achieves an accuracy of 98.1%. This demonstrates that our proposed approach can deal with a large variety of news pages. The state-of-the-art method proposed by Reis *et al.* [16] is based on template-level wrapper induction, and thus is unable to extract news articles from any other Web news site. Moreover, our approach achieves a significantly higher accuracy than the published accuracy of 87.71% in [16].

In Figure 16 we show a quite noisy news page from which our approach can still correctly extract the news title and the body. The extracted news article is shown in Figure 17.

Table 5: Testing results of the large scale experiment

Testing Site	Title Accuracy	Body Accuracy	News Accuracy
CNN	99.10%	99.60%	98.70%
BBC	99.70%	99.20%	98.90%
GC.CA	98.20%	98.20%	97.10%
YAHOO	100.00%	100.00%	100.00%
CBC	99.00%	99.50%	98.50%
CBSNEWS	99.30%	98.60%	98.30%
FOXNEWS	100.00%	100.00%	100.00%
NEWSWEEK	97.80%	100.00%	97.80%
SKY	100.00%	100.00%	100.00%
TIME	90.40%	100.00%	90.40%
USATODAY	99.80%	100.00%	99.40%
USNEWS	97.10%	98.10%	95.20%
Overall	**98.60%**	**99.40%**	**98.10%**

Figure 16: A quite noisy news page.

6. RELATED WORK

Our work is related to Web information extraction. In this section, we briefly review the related works.

Template-level wrapper induction is an important technique to extract data from pages generated from templates. Several automatic or semi-automatic wrapper induction methods have been proposed. The most representative ones are WIEN [11], SoftMeley [6], Stalker [14], RoadRunner [3], EXALG [1], TTAG [2], ViNTs [23]. We refer the readers to two surveys [4, 12] and two tutorials [13, 17] for more details related to information extraction and wrapper induction.

The methods mentioned above are generic methods. Designing a general method for Web information extraction is challenging due to the heterogeneity of Web content. Consequently, domain specific characteristics are often considered in effective and precise Web information extraction. One domain is Web news. The state-of-the-art news extraction method was proposed by Reis *et al.* [16], which is a template-level approach. First, a number of pages sharing the same kind of template are clustered to generate a specific extractor for that template. Given a news page P, the similarity between P and every existing wrapper is calculated using

Figure 17: The extracted news article.

the tree edit distance [18] as the similarity measure. The most similar wrapper is then selected to extract text passages from P. Last, some simple rules are used to find the title and the body among these texts. This template-level method has some serious limitations. First, in a news page, there tend to be many noises around a title, such as the subtitle, author names, story highlights and breaking news. These noises may be misidentified as the news title by some simple rules. Moreover, the method can only extract unformatted plain texts and is not user friendly.

All the template-level approaches cannot extract pages belonging to an unseen template until the wrapper for that template is generated. Moreover, any subtle changes of a template may invalidate a wrapper. Therefore, it is costly to maintain up-to-date wrappers for hundreds of websites.

Zheng *et al.* [25] proposed the V-Wrapper which showed some domain-level compatibility . The news title and contents are identified as some visual blocks and extracted accordingly. However, the importance of the news title is not fully recognized as the method does not discriminate features and models used to classify news titles and bodies. One drawback of this approach is that more than one title may be extracted from a single news page. For applications like news aggregation systems, identifying and extracting the unique title from a news page is highly desirable. Moreover, the block-based approach may fail to maintain the integrity of an extracted news article as it does not treat a news article as an inseparable unit. A block-level evaluation is used to measure the accuracy of news extraction, by which a page with misidentified title and fragmented news body extraction will still be rated with high accuracy if most of the blocks are correctly extracted. We evaluate our approach at the page-level. We consider how many news (both titles and bodies) are correctly extracted whereas V-Wrapper considers how many blocks are correctly extracted.

Web information extraction methods should be robust to the anomalies in various and numerous HTML pages [24]. A good browser can correctly build the DOM trees for the majority of ill-formed Web pages. Furthermore, a Web browser provides additional important properties from the pages, such as the spatial properties (i.e., the locations on the screen at which tags are rendered). These advantages have been well utilized for Web information extraction [8, 22]. In this paper we leverage these advantages of browsers to extract Web news.

7. CONCLUSIONS

In this paper, we propose an effective approach for learning a template-independent news wrapper using a very small number of news pages from a single news site. We extract not only plain texts of news as most of the current approaches do, but also extract images and animates within the news bodies. Furthermore, the extracted news articles will remain in the same visual style as in the original pages.

There are still many interesting problems need to be further studied. Currently this work assumes the given Web page to extract is indeed a news page. It would be interesting to investigate whether this work can be used to tell if the given page contains news article or not. Finally some noises might appear in the extracted article, e.g., ads close to the bottom of news bodies. Incorporating deep content analysis might be expected to help.

8. REFERENCES

[1] H. Arasu, A. Garcia-Molina. Extracting structured data from web pages. In *SIGMOD'03*, pages 337–348, 2003.

[2] J. Y. Chuang, S. L. Hsu. Tree-structured template generation for web pages. In *Web Intelligence'04*, pages 327–333, 2004.

[3] G. M. P. Crescenzi, V. Mecca. Roadrunner: Towards automatic data extraction from large web sites. In *VLDB'01*, 2001.

[4] S. Flesca. Web wrapper induction: a brief survey. *AI Communications*, 17(2):57–61, 2004.

[5] G. N. D. G. P. Gupta, S. Kaiser. Dom-based content extraction of html documents. In *WWW'03*, pages 207–214, 2003.

[6] M. T. Hsu, C. N. Dung. Generating finite-state transducers for semi-structured data extraction from the web. *Information Systems*, 23(8):521–538, 1998.

[7] Y. Hu, G. Xin, R. Song, G. Hu, S. Shi, Y. Cao, and H. Li. Title extraction from bodies of html documents and its application to web page retrieval. In *SIGIR'05*, pages 250–257, 2005.

[8] T. Irmak, U. Suel. Interactive wrapper generation with minimal user effort. In *WWW'06*, pages 553–563, 2006.

[9] N. Jindal and B. Liu. Review spam detection. In *WWW'07*, pages 1189–1190, 2007.

[10] K. A. M. C. Kamba, T. Bharat. An interactive, personalized, newspaper on the www. In *WWW'95*, 1995.

[11] N. Kushmerick. Wrapper induction: Efficiency and expressiveness. *Artificial Intelligence*, 118(1-2):15–68, 2000.

[12] B. A. d. S. A. S. T. J. S. Laender, A. H. F. Ribeiro-Neto. A brief survey of web data extraction tools. *ACM SIGMOD Record*, 31(2):84–93, 2002.

[13] B. Liu. Web content mining (tutorial). In *WWW'05*, 2005.

[14] S. K.-C. Muslea, I. Minton. A hierarchical approach to wrapper induction. In *Int. Conf. Autonomous Agents*, 1999.

[15] J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers*, 1999.

[16] D. C. Reis, P. B. Golgher, A. S. Silva, and A. F. Laender. Automatic web news extraction using tree edit distance. In *WWW'04*, pages 502–511, 2004.

[17] S. Sarawagi. Automation in information extraction and data integration (tutorial). In *VLDB'02*, 2002.

[18] S. M. Selkow. The tree-to-tree editing problem. *Information Processing Letters*, 6(6):184–186, 1977.

[19] B. C. T. K. L. Shen, H. T. Ooi. Giving meanings to www images. In *ACM MULTIMEDIA'00*, pages 39–47, 2000.

[20] R. Song, H. Liu, J. R. Wen, and W. Y. Ma. Learning block importance models for web pages. In *WWW'04*, 2004.

[21] V. Vapnik. Principles of risk minimization for learning theory. In *NIPS'91*, pages 831–838, 1991.

[22] B. Zhai, Y. Liu. Web data extraction based on partial tree alignment. In *WWW'05*, pages 76–85, 2005.

[23] W. W. Z. R. V. Y. C. Zhao, H. Meng. Fully automatic wrapper generation for search engines. In *WWW'05*, pages 66–75, 2005.

[24] W. Y. C. Zhao, H. Meng. Mining templates from search result records of search engines. In *SIGKDD'07*, pages 884–893, 2007.

[25] S. Zheng, R. Song, and J. Wen. Template-independent news extraction based on visual consistency. In *AAAI'07*, volume 22, pages 1507–1513, 2007.