Template-Independent News Extraction Based on Visual Consistency

Shuyi Zheng*
Pennsylvania State University
University Park, PA 16802
zhengshuyi@hotmail.com

Ruihua Song
Microsoft Research Asia
Beijing, 100080, China
rsong@microsoft.com

Ji-Rong Wen
Microsoft Research Asia
Beijing, 100080, China
jrwen@microsoft.com

Abstract

Wrapper is a traditional method to extract useful information from Web pages. Most previous works rely on the similarity between HTML tag trees and induced template-dependent wrappers. When hundreds of information sources need to be extracted in a specific domain like news, it is costly to generate and maintain the wrappers. In this paper, we propose a novel template-independent news extraction approach to easily identify news articles based on visual consistency. We first represent a page as a visual block tree. Then, by extracting a series of visual features, we can derive a composite visual feature set that is stable in the news domain. Finally, we use a machine learning approach to generate a template-independent wrapper. Experimental results indicate that our approach is effective in extracting news across websites, even from unseen websites. The performance is as high as around 95% in terms of F1-value.

Introduction

Since the birth of the Internet, Web information has continued to proliferate at an exponential pace due to ease of publishing and access. News is among the most popular interests for Web surfers. Some Web services, such as Google News and Yahoo! News, extract news from hundreds of sources. Their aim is to provide a better way for searching the latest news stories.

News articles are usually embedded in semi-structured format, e.g., HTML, as Figure 1 shows. To extract structured data from HTML pages, some tools and algorithms were reported in the literature on Web information extraction (Laender *et al.* 2002; Liu 2005). Among these approaches, the most traditional method is to generate a software or program, called wrapper, to extract data based on consistency of HTML DOM trees.

DOM tree-based approaches are template-dependent. The generated wrapper can only work properly for pages that share a specific template. Any change of a template may lead to the invalidation of a wrapper. Therefore, it is costly to maintain up-to-date wrappers for hundreds of websites.

When information sources expand to domain level, it is difficult (if not impossible) to induct a traditional DOM tree-based wrapper because pages from different websites lack structural consistency. First, similar content might be represented by different tags in pages from different Websites. To prove this, we did a statistical study on 295 Web news pages and found more than 10 types of tags used to wrap news headlines. The tags include <B>, <H1>, <FONT>, and <SPAN>. Second, DOM trees generated by different templates can have entirely different topological structures. For instance, news content might appear at different levels of DOM trees. We have also performed a statistical experiment on the same 295 Web page set. The results indicate that the depth of news content distributes in a wide range (from two through 29). To demonstrate these points, we compare two news pages (Figure 1) whose DOM trees are shown in Figure 2(a) and Figure 2(b). As the figures illustrate, the DOM trees differ in terms of tag and topological structures.

Despite the structural dissimilarities among pages of different templates, people can still easily identify a news article from a news page at a glance. For example, when people browse the two news pages in Figure 1, they can quickly locate the news articles without any difficulty, even in the case in which the news is written in a language they may not understand (e.g., Arabic news page).

In this paper, we discuss why human beings can identify news articles regardless of templates. We propose a novel news extraction approach that simulates human beings. The approach is template-independent, because it is based on a more stable visual consistency among news pages across websites. First, we represent a DOM node with a block and basic visual features to eliminate the diversity of tags. Second, extended visual features are calculated as relative visual features to the parent block. Such features can handle the dissimilarity of topological structure. Third, we combine two classifiers into a vision-based wrapper for extraction. Experimental results indicate the proposed approach can achieve high extraction accuracy as about 95% in terms of F1-value which substantially outperforms exisiting techniques.

The remainder of this paper is organized as follows. We first describe our proposed approach. Then experimental results are reported, which is followed by a brief review of related work. Finally, we draw conclusions.

*Work was done when the author was visiting Microsoft Research Asia.

(a) from ABCNews.com (in English)

(b) from News.BBC.co.uk (in Arabic)

Figure 1: Two sample news pages

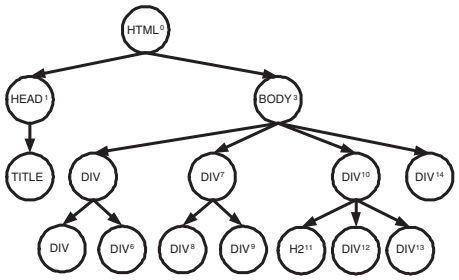
Our Approach

We target deriving domain level visual consistency by using visual features. We argue that visual consistency can lead to high extraction accuracy even though template consistency may be missing.

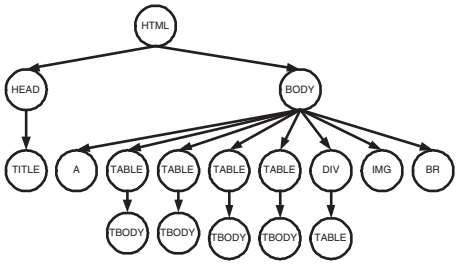
Visual Consistency

When people browse a Web page, they are subconsciously guided by the experience they have accumulated in browsing other similar pages in similar domains. In the case of news, people commonly seek a block with the following visual features: (1) Its area is relatively larger than other page objects around it. (2) At the top of the block, there is usually a bold-faced line which is the news headline. (3) It is mostly occupied by contiguous text paragraphs, sometimes mixed with one or two illustration pictures. (4) Its center is close to that of the whole page. . . . They would never go for a slim and long block because they assume that page object in that shape is unlikely to contain the core information.

This is understandable because when a Website developer designs Web pages in a specific domain, usually some hidden conventions of a domain that have already been accepted by most people are followed. For instance, designers generally do not put main content at the corner of a page, occupying a small portion of the page area. We notice that such design conventions are irrelevant to the content text or HTML tags. It is a kind of visual level convention that leads to the visual consistency among all pages in the same domain.



(a) for page in Figure 1(a)



(b) for page in Figure 1(b)

Figure 2: DOM trees for pages in Figure 1

Data Representation

In our solution, all DOM nodes are treated as rectangular blocks. HTML tags are not used to distinguish different blocks. Instead, we use a group of visual features to represent them and determine the similarity among them.

Definition 1 *(Visual Block) A visual block is a visible rectangular region on Web page with fixed position and nonzero size. It is rendered from a pair of HTML tags, e.g.,* `<DIV>` *and* `</DIV>`*, or text between them.*

For both coding and description convenience, we assign a unique identity for each block within a Web page, called *Block ID*.

Every Web page can be converted to a visual tree. Again, we take news page in Figure 1(a) for instance. Its block structure is displayed in Figure 3 (Blocks at deeper levels are omitted). The superscript numbers (Block ID) of both figures indicates the relations between HTML tags and blocks rendered by them. As we can see, the whole page can be considered as the biggest block (called page level block) rendered by `<BODY>` and `</BODY>`. Note that although each visual block is rendered by a pair of HTML tags, such tag information is not used as a feature for leaning our wrapper.

Given an HTML source, an HTML rendering algorithm or an HTML parsing tool can be used to obtain such a visual tree. In our implementation, we choose the Microsoft MSHTML library [1] to render and parse HTML source. The MSHTML library provides interfaces to access visual information for any DOM nodes. Based upon these visual cues,

[1] More details about MSHTML library can be found at: http://msdn.microsoft.com/workshop/browser/editing/mshtmleditor.asp

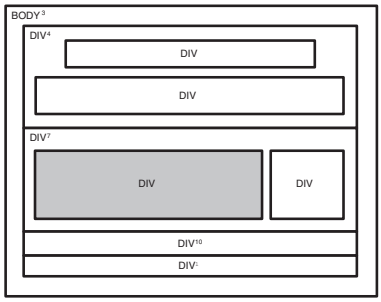


Figure 3: Visual blocks of page in Figure 1(a)

we derive needed visual features for our news domain application. The basic visual features are listed below in 4 categories.

1. **Position features**: Left, Top (the coordinate of left-top corner of a block), NestedDepth;
2. **Size features**: Width, Height;
3. **Rich format features**: FontSize, IsBoldFont, IsItalicFont;
4. **Statistical features**: ImageNumber, HyperlinkNumber, TextLength, ParagraphNumber, ItalicParagraphNumber, BoldPargraphNumber, TableNumber.

Consequently, the tag diversity of DOM trees does not affect our algorithm, even at the domain level. For example, the title in Figure 1(a) is wrapped by <SPAN> and that in Figure 1(b) is wrapped by <FONT>. For most T-Wrapper induction methods, the corresponding DOM nodes of the two titles cannot match each other. In our approach, they are considered visually similar because they are both boldfaced with a high ratio of width and height.

Extended Visual Features

Since the topological structures of DOM trees are quite different at domain level, they are not used for matching purposes. Instead, we use "parent-child" visual relations between two blocks.

Definition 2 *(Parent Block and Child Block) Given two visual blocks b_1, b_2, we say b_1 is b_2's parent block (or b_2 is b_1's child block), iff b_1 covers b_2 and these exist no block b_3 in the same page, where b_1 covers b_3 and b_3 covers b_2.*

Based on this definition, we can define a set of extended visual features to represent relations between a child block and its parent block. E.g.,

$$RelativeWidthOfParent = Width/ParentWidth$$

$$RelativeLeftOfParent = Left - ParentLeft$$

The primary difference between "parent-child" visual relation and topological structure of DOM tree is that it has no constraints on path depth. More specifically, for each DOM node, its position in DOM tree is determined by the tag path from root node to itself. This tag path plays an important role in most traditional DOM tree-based methods. In our approach, the "parent-child" visual relations can be used to match any two blocks as to paternity in spite of their nested depth. In short, topological diversity of DOM trees can also be handled effectively in our approach.

Learning a Vision Based Wrapper

Based on the visual consistency, we target the creations of a robust vision-based wrapper for an entire domain. For statement convenience, we denote our vision-based generated wrappers as *V-Wrapper*, whereas traditional DOM tree-based wrappers as *T-Wrapper*.

Actually, when people browse a Web page, they subconsciously combine various visual features with different priorities and weights in seeking target information. If we can simulate this human behavior, we can likely derive a composite visual feature which is stable enough to form a domain-level visual consistency. In our system, we use Adaboost (Freund & Schapire 1997) to learn the composite visual feature. We chose Adaboost because it is an effective algorithm in weighting different features.

Like traditional wrapper induction methods, our approach also requires a set of manually labeled pages for the purpose of training. In our work for news domain, we define two label sets for inner block and leaf block.

Definition 3 *(Inner Block and Leaf Block) A block is an inner block iff it has at least one child block; otherwise, a leaf block.*

For a leaf block, annotators give labels of news *Title* or *Content* to relevant blocks while other leaf block is inferred as the label of *Others*.

$$\mathcal{L} = \{Title, Content, Others\}$$

For an inner block, labels are derived.

Theorem 1 *A inner block is treated as a positive block iff at lease one of its child block is labeled rather than Others.*

If we consider positive inner blocks (PI), negative inner blocks (NI) as two derived labels.

$$\mathcal{L}^{'} = \{PI, NI\}$$

Actually, a label is not only used in training pages, but also in testing pages to present the extraction result. We name the former pre-label and the latter post-label for differentiation purpose. In our experiments, by comparing the post-label with corresponding pre-label, we can easily evaluate extraction accuracy in terms of *precision*, *recall* and *F1-Value*.

To learn a V-Wrapper, we need to simulate human behavior involved in browsing news pages. Actually, we can think of this behavior as a two-step process. The first step is to roughly locate the main block based on all kinds of visual features presented in the page and excluding all irrelevant content like advertisements. The second step is to look into the main block more carefully and identify which is the headline, the news body, and so on. These two steps normally require different visual features.

Similiarly, we developped a two-step learning process for V-Wrappers.

In step 1, we select all inner blocks B_1 with their corresponding derived labels, and train a classifier as to whether an inner block contains some pieces of news.

In step 2, we choose all leaf blocks whose parent blocks are predicted as *PI* by the classifier in step 1. These leaf blocks compose B_2. Then, similarly, we train a classifier to predict which label a leaf block matches.

News Extraction using a V-Wrapper

The extraction algorithm using generated V-Wrapper for news domain also have two steps:

1. The first step seeks to extract leaves blocks whose parents are positive inner blocks as candidates for target information.
   The algorithm (See Figure 4) is a recursive Top-Down process. It starts from the biggest block (page block) and stops at leaf blocks or negative inner blocks. We do not continue to deal with child blocks of negative inner block b because all blocks of the subtree rooted at b are considered as negative blocks, due to Theorem 1.
2. The second step is to label different types of information from candidates blocks obtained in the first step.
   In this step, the leaf block classifier is used to match each candidate block with labels. The output of this step is a post label $\mathcal{L}(p)$ for test page p. It indicates the target information which must be extracted from p.

Algorithm: Identify_Candidate_Blocks(p)
1. **begin**
2. Parse p to visual block set B;
3. $pageBlock :=$ page level block in B;
4. $plBlocks :=$ empty block set;
5. Extract($pageBlock, plBlocks$) ;
6. **return** $plBlocks$;
7. **end**

Algorithm: Extract($b, plBlocks$)
1. **begin**
2. **if** b is a inner block **then**
3. **if** b is labeled as *PI* **then**
4. $childBlock :=$ the first child block of b;
5. **while** $childBlock$ is not NULL
6. Extract($childBlock, plBlocks$);
7. $childBlock := childBlock$'s next sibling;
8. **endwhile**;
9. **endif**;
10. **else**
12. Add b to $plBlocks$;
14. **endif**;
15. **end**

Figure 4: Algorithm for identifying candidate blocks

Experiments

The experiments in this paper are designed to demonstrate the performance of our V-Wrapper induction system, as well as to show the advantages of V-Wrapper in contrast with traditional T-Wrapper.

To compare it with traditional methods, we have also built a T-Wrapper generation system using state-of-the-art techniques (Tree Edit Distance, Regular Expression Inference and Sequence Alignment). Our T-Wrapper generation system adopts the main ideas from Reis et al.'s work (Reis *et al.* 2004) and Chuang et al.'s work (Chuang & Hsu 2004). Their methods are close to ours and proved to be effective on template level T-Wrapper induction.

Experiment Setup

We collected 295 pages from 16 online news sites to use in our experiments. For comparison's sake, we only collected pages sharing one common template for each site. i.e., in our experiments, site level equals template level. Pages of each site are randomly divided into three groups to perform cross validation for all experiments. More specifically, we conducted each experiment three times by taking different page groups as test sets for each site. Then we averaged three results as the final result.

All pages are manually labeled for extraction result evaluation. Suppose the extraction result of a test page p is rendered in the format of post label $\mathcal{L}_{post}(p)$ and its corresponding manual label is $\mathcal{L}_{pre}(p)$, we can compute three measures to evaluate the extraction accuracy of certain extraction type τ: *precision*, *recall* and *F1-Value*

$$precision = \frac{|\mathcal{L}_{pre}(p) \cap \mathcal{L}_{post}(p)|}{|\mathcal{L}_{post}(p)|} \quad (1)$$

$$recall = \frac{|\mathcal{L}_{pre}(p) \cap \mathcal{L}_{post}(p)|}{|\mathcal{L}_{pre}(p)|} \quad (2)$$

$$\textit{F1-Value} = \frac{2 \cdot precision \cdot recall}{precision + recall} \quad (3)$$

where $|\mathcal{L}|$ is the number of items labeled as τ in $\mathcal{L}$.

We implemented two groups of experiments on a PC with 3 GHz Pentium 4 processor and 1015MB RAM.

Experiment One: Domain Level Compatibility

The objective of the first experimental group is to demonstrate what we call domain-level compatibility of V-Wrapper. It is carried out as follows:

First, we take one site training set as the initial training set TrS_1, and a V-Wrapper W_1 can be learned from TrS_1. Then we add another site training set to TrS_1 and get a new training set TrS_2 which generates W_2. Repeat this step until all the training sets of the 16 sites are used to generate W_{16}. Thus, we can get 16 different V-Wrappers. We use these 16 V-Wrappers to extract news text from same test set TeS, which is the combination of 16 site test sets.

Figure 5 shows changes of extraction accuracy as the training sets increase. Initially, the first V-Wrapper learned from one site's training set can only achieve 50.93% accuracy in terms of F1-Value. After utilizing three sites training

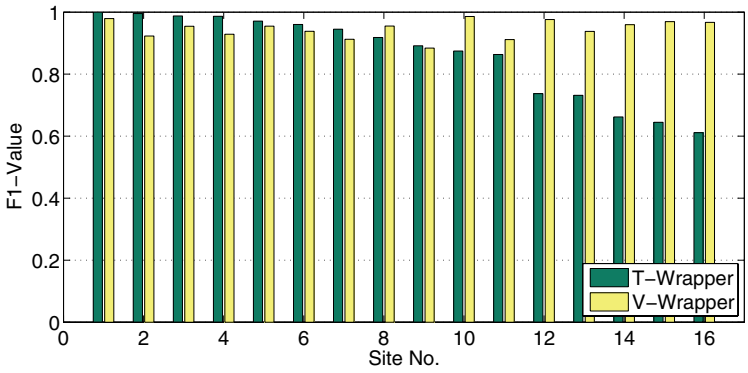


Figure 6: T-Wrapper vs.V-Wrapper: Site By Site

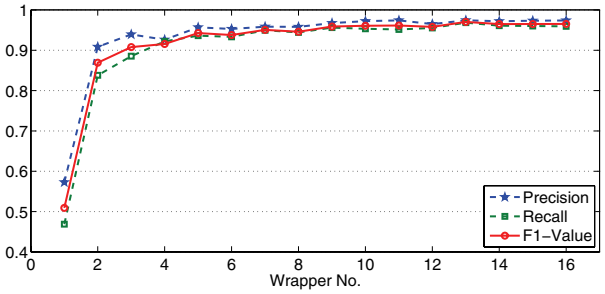


Figure 5: Result of experiment one

sets, the F1-Value stays above 90% and becomes quite stable. This adequately proves the domain level compatibility of V-Wrapper generated by our approach. We can imagine that given another news Website, our V-Wrapper can still achieve optimal extraction accuracy without re-training.

Experiment Two: V-Wrapper vs. T-Wrapper

The second group of experiments are designed to compare V-Wrapper with T-Wrapper site by site. Since V-Wrapper is domain level wrapper, we only generate one V-Wrapper from all site training sets and apply it to all test sets. This wrapper is same as W_{16} used in Experiment One. On the contrary, T-Wrapper is site level wrapper, every site generates a T-Wrapper and use it to test corresponding test set. The average F1-Value is 94.96%.

Figure 6 shows the comparison results of V-Wrapper and T-Wrapper. As displayed in Figure 6, although T-Wrapper outperforms V-Wrapper on several sites, even achieve 100% accuracy on NYTimes.com, its average extraction accuracy is not as good as V-Wrapper (F1-Value: 88.32%). This can be explained by two reasons.

First, T-Wrapper's performance highly depends on DOM structure consistency of each site, mainly in DOM tree structure. If pages in a site are very similar (e.g., NYTimes.com), T-Wrapper can achieve almost perfect extraction results. However, if pages in a site are diverse in structure (e.g., News.Yahoo.com), T-Wrapper's extraction accuracy drops quickly.

Second, as stated early, it is difficult to decide the size of training sets for T-Wrapper induction; thus, if the randomly selected training set is not large enough, the T-Wrapper probably cannot cover all pages in the test set. To prove this, we have performed an experiment for case study purpose on FT.com which has the lowest extraction accuracy in all test sets. We use pages in test set to refine the generated T-Wrapper and implement extracting process again. We find that the extraction accuracy in terms of F1-Value obtains an improvement from 61.11% to 94.25%).

In contrary to T-Wrapper, V-Wrapper's performance is pretty stable in the whole domain. Most errors of V-Wrapper are caused by noise information (e.g., copyright information), which is visually similar to news text.

Related Work

Our work belongs to the area of Web Information Extraction (IE). It has received a lot of attention in recent years. We group all related works into two categories and briefly describe each class as follows. We refer the reader to a good survey (Laender *et al.* 2002) and two tutorials (Sarawagi 2002)(Liu 2005) for more works related with IE.

Extracting structured data in template level: The most popular technique for this category is wrapper induction. Several automatic or semi-automatic wrapper learning methods have been proposed. Among them, WIEN (Kushmerick, Weld, & Doorenbos 1997) is the earliest method as we know on automatic wrapper induction. Other representatives are SoftMeley (Hsu & Dung 1998), Stalker (Muslea, Minton, & Knoblock 1999), RoadRunner (Crescenzi, Mecca, & Merialdo 2001), EXALG (Arasu & Garcia-Molina 2003), TTAG (Chuang & Hsu 2004), works in (Reis *et al.* 2004) and ViNTs (Zhao *et al.* 2005).

TTAG and works in (Reis *et al.* 2004) are close to our T-Wrapper induction system used in our work for comparison purpose. TTAG proposes a top-down level-by-level approach to generate a tree-structured wrapper. A dynamic programming method is used in each level's alignment. In (Reis *et al.* 2004), tree edit distance is used to generate extraction patterns from clustered pages. Pages in each cluster share a common template. Though they also achieve opti-

mal experimental results on news Websites, the approach is template-dependent.

ViNTs (Zhao *et al.* 2005) uses some visual features in generating wrappers for search engine results. In the sense of using visual cues, it is advantageous when contrasted with previous work. However, the visual features used in ViNTs are only limited to the content shape-related features. They use them to help identify the regularities between search result records. Therefore, ViNTs still depends on structural similarities and must generate a wrapper for each search engine.

Vision assisted techniques: The problem of judging the importance or function of different parts in a web page attracts a lot of attentions. Some proposed approaches (Kovacevic *et al.* 2002; Song *et al.* 2004; Yin & Lee 2004; 2005) take some advantages of visual information more or less. Kovacevic et al. (Kovacevic *et al.* 2002) used visual information to build up an M-Tree, and further defined heuristics to recognize common page areas such as header, left and right menu, footer and center of a page. Some other works try to decide block importance levels or functions in a single web page, by learning algorithm (Song *et al.* 2004) or random walk models (Yin & Lee 2004; 2005). In these works, visual features, such as width, height, position, etc., are reported useful.

Conclusion

In this paper, we proposed a template-independent news extraction approach based on visual consistency. We first represent a page as a visual block tree. Then, by extracting a series of visual features, we can derive a composite visual feature set that is stable in the news domain. Finally, we use a machine learning approach to generate a vision-based wrapper. Once a V-Wrapper is learned, it is able to deal with all pages of the domain without re-training. Thus, our approach saves a lot of time and money for wrapper maintenance. Experimental results demonstrate that the generated V-Wrapper not only has a beneficial domain compatibility, but also can achieve extraction accuracy of as much as 95% in terms of F1-value. As future work, we plan to broaden our application to other domains.

Acknowledgments

We would like to thank Chris Quirk for providing data and valuable discussions. We are grateful to Dwight Daniels for detailed edits on writing this paper. Comments from the four anonymous referees are invaluable for us to prepare the final version.

References

Arasu, A., and Garcia-Molina, H. 2003. Extracting structured data from web pages. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, 337 – 348.

Chuang, S.-L., and Hsu, J. Y.-j. 2004. Tree-structured template generation for web pages. In *Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence*, 327 – 333.

Crescenzi, V.; Mecca, G.; and Merialdo, P. 2001. Roadrunner: Towards automatic data extraction from large web sites. In *Proceedings of the 27th International Conference on Very Large Data Bases*, 109 – 118.

Freund, Y., and Schapire, R. E. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences* 55(1):119–139.

Hsu, C.-N., and Dung, M.-T. 1998. Generating finite-state transducers for semi-structured data extraction from the web. *Information Systems, Special Issue on Semistructured Data* 23(8):521–538.

Kovacevic, M.; Diligenti, M.; Gori, M.; and Milutinovic, V. 2002. Recognition of common areas in a web page using visual information: a possible application in a page classification. In *Proceedings of the 2002 IEEE International Conference on Data Mining*, 250 – 257.

Kushmerick, N.; Weld, D. S.; and Doorenbos, R. B. 1997. Wrapper induction for information extraction. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 729–737.

Laender, A. H. F.; Ribeiro-Neto, B. A.; da Silva, A. S.; and Teixeira, J. S. 2002. A brief survey of web data extraction tools. *SIGMOD Record* 31(2):84–93.

Liu, B. 2005. Web content mining (tutorial). In *Proceedings of the 14th International Conference on World Wide Web*.

Muslea, I.; Minton, S.; and Knoblock, C. 1999. A hierarchical approach to wrapper induction. In *Proceedings of the 3rd Annual Conference on Autonomous Agents*, 190 – 197.

Reis, D. C.; Golgher, P. B.; Silva, A. S.; and Laender, A. F. 2004. Automatic web news extraction using tree edit distance. In *Proceedings of the 13th International Conference on World Wide Web*, 502 – 511.

Sarawagi, S. 2002. Automation in information extraction and data integration (tutorial). In *Proceedings of the 28th International Conference on Very Large Data Bases*.

Song, R.; Liu, H.; Wen, J.-R.; and Ma, W.-Y. 2004. Learning block importance models for web pages. In *Proceedings of the 13th International Conference on World Wide Web*, 203 – 211.

Yin, X., and Lee, W. S. 2004. Using link analysis to improve layout on mobile devices. In *Proceedings of the 13th International Conference on World Wide Web*, 338–344.

Yin, X., and Lee, W. S. 2005. Understanding the function of web elements for mobile content delivery using random walk models. In *Special interest tracks and posters of the 14th International Conference on World Wide Web*, 1150–1151.

Zhao, H.; Meng, W.; Wu, Z.; Raghavan, V.; and Yu, C. 2005. Fully automatic wrapper generation for search engines. In *Proceedings of the 14th International Conference on World Wide Web*, 66 – 75.