Site-Independent Template-Block Detection

Aleksander Kołcz[1] and Wen-tau Yih[2]

[1] Microsoft Live Labs, Redmond WA, USA
[2] Microsoft Research, Redmond WA, USA

Abstract. Detection of template and noise blocks in web pages is an important step in improving the performance of information retrieval and content extraction. Of the many approaches proposed, most rely on the assumption of operating within the confines of a single website or require expensive hand-labeling of relevant and non-relevant blocks for model induction. This reduces their applicability, since in many practical scenarios template blocks need to be detected in arbitrary web pages, with no prior knowledge of the site structure. In this work we propose to bridge these two approaches by using within-site template discovery techniques to drive the induction of a site-independent template detector. Our approach eliminates the need for human annotation and produces highly effective models. Experimental results demonstrate the usefulness of the proposed methodology for the important applications of keyword extraction, with relative performance gain as high as 20%.

1 Introduction

The Web has become the most important information resource. Numerous applications and services for information retrieval and content extraction have been created to analyze web documents automatically in order to serve high-quality, relevant information to millions of users daily. The effectiveness of these applications relies on their ability to identify the "important" portions of a web page and separate them from other items that may be present, such as navigation bars, advertisements, etc. Detecting these *template* or *noise* blocks has thus become a major preprocessing step in several practical web applications [14][4]. In this paper, we propose a robust and effective approach to detect the template blocks. In contrast to many existing approaches, our method operates in a site-independent fashion, where information such as site structure is not needed. It is trained using a large collection of data sampled from various websites. Reliable labels are gathered automatically based on statistics of the characteristics of in-domain pages, which completely eliminates the expensive process of human annotation. This makes it very attractive for many application scenarios. Our approach not only achieves high accuracy in its own task – template block detection, but also brings significantly positive impact on the application of *keyword extraction*.

J.N. Kok et al. (Eds.): PKDD 2007, LNAI 4702, pp. 152–163, 2007.

2 Prior and Related Work

When analyzing the content of a web-page, the hypertext is usually first represented as a DOM tree. The goal of web-page cleaning is then to postprocess it by either removing certain element nodes or by assigning different importance weights to different blocks. Both block elimination [14] and block weighting [13] have been shown to improve the accuracy of document clustering and classification. The differences between various solutions proposed in the literature lie in the areas outlined below.

2.1 Segmentation Heuristics

Not all HTML tags impact the block appearance of a web page. Domain knowledge is typically used to identify the subset of possible HTML tags (e.g., table elements) that are considered as possible block separators [4], although not all such nodes correspond to clearly identifiable rectangular regions of a page. [2] restrict valid block nodes to such that, when rendered, are visually separated from other regions by horizontal and vertical lines and, in addition, have content that is sufficiently cohesive. Although in most cases a block will contain some text and clickable links, in applications such as advertisement removal, a block may be defined as an area in the HTML source that maps to a clickable image [9].

2.2 The Same-Site Assumption

When approaching the problem of noise block detection, many researchers have taken the "same-site" assumption, i.e., restricted the problem to the confines of the same website, where it is likely that web pages were generated using a set of common templates or layout styles [1,4,14,7]. In particular, whether a block should be considered as part of the template can be naturally defined by its relative block frequency. In [1], a shingling algorithm was used to compute robust fingerprints of block content. These fingerprints were then applied to identify near-duplicate blocks on pages belonging to the same site with some additional constraints on the nature of a block and the set of pages for which block occurrences were counted. In [7], rather than computing block similarity via text shingling, a direct MD5 digest of the HTML snippet corresponding to a block was computed and the relative frequency of such fingerprints in the collection of pages of a site were counted.

Judging block similarity by a fingerprint match may be brittle. A number of authors considered comparing pairs of blocks directly. [14] compressed pages from the same site into a site tree, where nodes from different pages are considered equivalent if they have the same position in the DOM tree and the same attributes. However, nodes can be merged also when the textual content is sufficiently similar even if the HTML tags/attributes are different. Block-to-block similarity computations are thus limited to nodes in related positions in their DOM trees. A similar approach was taken by [4], where all blocks of all pages are compared for similarity using both textual and structural features. The computational complexity is reduced by keeping track of corresponding blocks in pages

that had already been processed. [3] proposed an approach that scales well to large data collections. Here, blocks are first grouped into clusters of individual styles and then further grouped by content similarity within each cluster.

2.3 Site-Independent Relevant/Noise Block Identification

In practical applications, simple site-independent heuristics are often applied to remove top/bottom banner ads and boiler-plates. In [9], a rule-based system is proposed to extract banner ads, although this is a problem narrower than noise-block detection. In [4], blocks are categorized as text-based, image/link based etc., depending on the ratio of one particular type of content within a block to all others, which leads to different weights of different features. The feature weights are then compared among the blocks belonging to a single page to identify, for example, the largest text block. Block attributes were also used in [4] in conjunction with labeled data to train an inductive model which showed good performance in identifying the relevant blocks using a small set of 250 pages. Song *et al.* [10] took an approach relying on a two-dimensional layout of a web page. It was shown that the combination of positional and non-positional features can lead to very accurate predictive models given a classifier such as an SVM and a quantity of labeled data.

3 Bridging the Gap Between Site-Dependent and Site-Independent Approaches

The results of published research in the noise-block detection area can be summarized as follows:

- It is possible to automatically detect template blocks within pages of the same site.
- Using hand-labeled data corresponding to noise and non-noise blocks in pages from multiple sites, it is possible to build accurate inductive models estimating the importance of a block on a page.
- Removal or down-weighting of content in template blocks tends to increase the accuracy of data mining tasks such as classification or clustering.

One notes that although the template discovery process for a single site is automatic, building site independent noise detection models has so far relied on expensive hand labeling of all blocks within a page for a training corpus. Our contribution is to apply automatic template discovery to generate large quantities of labeled data for use in building the inductive models. We hypothesize that to achieve high accuracy one does not need to acquire labels for all blocks within a training web-page. Given a large quantity of web pages, we propose to pool only those blocks from individual pages that can be confidently identified as either a template or a non-template. The lack of a hand-labeled truth set might be seen as preventing evaluation of the quality of the resulting model. We take the position, however, that in practical applications, the quality and impact

of web-page cleaning should be primarily measured by the secondary metrics of how the accuracy of the data mining tasks is affected.

Our approach can be described as follows (the algorithm settings used in our experiments are further detailed in Section 4):

1. A sample of pages is collected from sites that have substantial number of pages, which enables application of site-specific template discovery techniques. A variety of sampling regimens could be considered. Traditionally, uniform sampling of pages produced by a webcrawl of a particular site tends to be used. However, this does not guarantee fair access to all site content, due to the implementation details of any particular crawler used and the fact that many sites contain dynamic content. An alternative might be to sample from a web-log pertaining to an actual site, which focuses on pages that are actually getting accessed.
2. A block segmentation algorithm is applied to each page and block frequency statistics within each site are computed. Since our goal is to identify template blocks within the set of pages sampled from a site rather than building a template detection model for that site, a simple fingerprint-based technique (adapted from [7]) is used. The block identification algorithm is constrained to HTML elements that are typically rendered in the form of a visual block. Additionally, a valid block needs to contain sufficient amount of textual content.
3. Blocks having sufficiently high document frequency are considered as templates. Unique blocks are considered as non-templates. The remaining blocks are ignored. This provides a labeled set of blocks for a particular site.
4. The feature vectors for labeled blocks extracted from the collection of sites are aggregated into a dataset used to train an inductive template detection model. We consider features describing a block in isolation from the remainder of the page (e.g., its textual content including length and word count, punctuation and formatting of text, the presence and frequency of HTML tags, with separate counting of image and anchor tags), features capturing the relative difference of the block from the remainder of the page (KL divergence between textual contents, relative word and tags counts), as well as features capturing the position of the block node within the DOM tree (e.g., line number, distance from the root, distance from the left-most sibling). The feature generation process aims to introduce a rich representation of potentially correlated attributes.
5. Given the training data, a model is induced and subsequently applied in a site-independent manner on a page-by-page basis. A number of machine learning techniques can be used in modeling. Section 4 discusses the techniques chosen in this work in more detail.

While block frequency counting can lead to fairly confident identification of template blocks, assuming that the ones that are very infrequent are non-templates may sometimes be inaccurate. An infrequent block could for example be a block in a new format that has not yet been widely deployed, or a result of

a copy and paste of the block, or the entire page, from somewhere else. To limit the effect of class-noise in our system, blocks with frequency greater than one but smaller than a frequent-threshold cutoff point are ignored. Since the template blocks are repetitive, for any web site many more examples of non-template blocks can be acquired. To curb the amount of imbalance, in the hierarchical processing of DOM trees we retain only the top-most non-template block that decomposes in non-template blocks only (i.e., all of its sub-blocks have the site frequency of 1).

4 Experiment Details

We used two independent sets of data (one that was used in [7] and one derived from the Open Directory Project (ODP)[1]) to learn template detection models. The motivation for using two alternative sets was to assess the importance of a particular sampling regimen on the usefulness of the resulting model. The two datasets differ drastically in the sampling philosophy. One uses a uniform sample from the data generated by a large-scale web-crawl and one uses a human generated collection of links, with unknown biases as to what areas of a site are being accounted for.

4.1 Template Detection Using the Crawl-Sample Dataset

The data for this experiment corresponded to the results published in [7], where a random sample of domains was extracted from a large web crawl (of approx. 2,000,000,000 pages) and subsequently pruned to retain only those sites having at least 200 pages. This resulted in a dataset containing 109 sites and 21,349 pages. For each page, the HTML content was parsed into a DOM tree from which nodes corresponding to invalid HTML elements or invalid attributes of valid elements were removed. Comment and script sections were also eliminated. A potential block node had to satisfy the following two requirements:

- It had to correspond to a predefined set of HTML tags that typically affect visual block layout {`blockquote, dd, div, dl, dt, h1, h2, h3, h4, h5, h6, li, ol, pre, small, table, td, th, tr, ul`}.
- It had to have sufficient text content. For each node, the underlying text was extracted and processed to remove the leading and trailing space as well as to collapse consecutive whitespace into one. A block was then considered to be a candidate if the length of the normalized text content was at least 40 characters and at least 3 unique words.

When processing web pages belonging to the same site, potential block nodes on each page were selected and for each one the page count of the node was recorded. A node was identified by an MD5 digest of its textual content, so that nodes having different markups but resolving to the same text were treated as

[1] http://www.dmoz.org

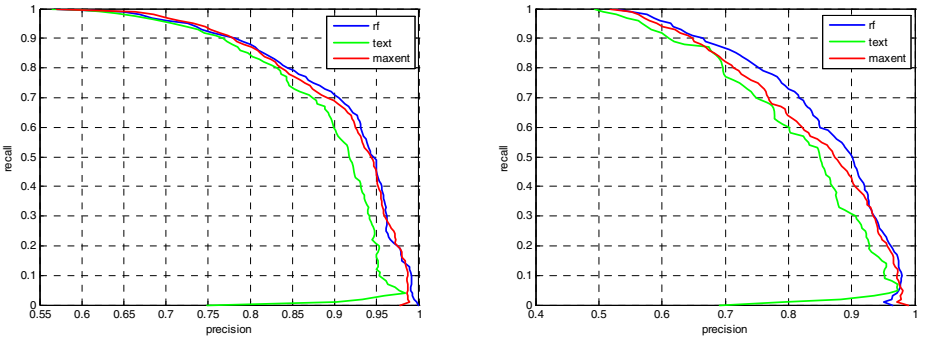

Fig. 1. Template detection performance over the web-crawl dataset (left) and the ODP dataset (right) in terms of the precision-recall curve. Using text-only features performs well (using Naive Bayes, labeled as *text*), but further improvements are possible by considering non-textual attributes (using logistic regression, labeled as *maxent*). Random forest (labeled as *rf*) has an advantage over logistic regression, which is more pronounced for the ODP dataset.

synonymous. In the hierarchical DOM structure, if a node was found to have too little textual content, it and all its descendants were subsequently ignored. Similarly, duplicates were also ignored. In the end, for each site a frequency distribution of the textual fingerprints was obtained and, following [7], a node was declared to be a template if it was present in at least 10% of the site's pages. A node was declared a definite non-template candidate if it occurred only on a single page.

Given the identity of template and non-template nodes, the data corresponding to each site were processed again to extract feature sets out of each node. Additionally, in this pass, a node was considered a non-template if it did not contain any template nodes or frequent nodes underneath its node hierarchy. This prevented, for example, the full web-page to be considered as an example of a non-template. The essence was to capture the positional information of a block in the DOM tree and in the page source, as well as the content and markup properties of a block.

The positive and negative example feature vectors due to individual sites were grouped together. In a 10-fold *cross-site* cross-validation experimental procedure, in 10 runs the data corresponding to 10% of sites were used for testing with the remaining data used to build a predictive model. Note that we *did not* use data from the same site for both training and testing, and there was no known relationship between different sites.

For each cross-validation round the training set was deduplicated and then the training data underwent MDL-based discretization [5] and was converted to binary features. Again, training nodes with the same feature vectors were deduplicated. The same discretization model was applied to the test data, although no deduplication was performed in this case.

The feature vectors consisted of two main components: the subset corresponding to word features and the subset corresponding to all other features, with text attributes being much more numerous. Given that natural sparsity of text, we decided to split the modeling into building a text based classifier first and using its output score as a feature to be combined with other features in inducing the final template-block classifier. For text classification we used a version of Naive Bayes (labeled as *text*). For the overall classifier, we considered comparing logistic regression (labeled as *maxent*) as the linear model and random forest (labeled as *rf*) as the non-linear model. Figure 1 (left) shows the precision-recall curves for the classifiers considered. At the relatively high precision of 90%, it is possible to achieve recall of 70%, but since the non-template data did not undergo hand-labeling it is possible that precision is actually higher. This can be considered as indicative of high effectiveness, especially given that the data from the sites over which the classifiers are tested were not used in their induction. Note that text features alone tend to be a good predictor, but combination with other attributes leads to substantial improvement, particularly in the area of high precision. Logistic regression performs comparably to a random forest, although the latter appears to have an advantage in the region of precision between 80% and 95%.

4.2 Template Detection Using the ODP Dataset

A uniform sample of a web crawl is not always easy to obtain. On the other hand, various biased samples of the web (collections of links, directories, etc.) are readily available. A question arises to what extent a useful template block detection model can be defined using such biased samples. In order to answer this question, we used the Oct 6, 2006 snapshot of the Open Directory Project (ODP). We sorted the sites according to their ODP page presence and considered the ones having at least 200 pages in the directory. In cases where the number of pages listed was greater, we took a random sample of 200. After retaining only valid unique HTML documents, the dataset consisted of 21,869 web pages corresponding to 131 different sites. This dataset was used to derive a template-block model (the ODP template model) using methodology and settings identical as in the experiment described in Section 4.1. The accuracy of template detection using these data is summarized in Figure 1 (right). Qualitatively, the results are similar to those obtained using the web-crawl data, although the detection rates at any given precision appear to be lower. This could be attributed to the bias of ODP "sampling", which might have resulted in more true-template blocks being considered as non-templates. Also, note that for this dataset the benefit of using non-linear random forest rather than logistic regression appears to be more pronounced. All in all, even with non-uniform sampling, the proposed template detection methodology appears to be quite effective.

5 Application to Cross-Domain Keyword Extraction

Noise-block detection has previously been found beneficial to traditional information retrieval tasks such as document ranking. Here we examine if it can help

in the relatively recent application of *keyword extraction* [6,11,12,8,15], the fundamental technology for content-targeted advertising such as Google's AdSense program and Yahoo's Contextual Match product.

Given a target web document, the goal is to identify keywords that are relevant to the content. The output keywords can be used to match keywords advertisers bid on. Corresponding advertisements are then displayed on the target page, which is supposed to be relevant to the content. Keywords contained just in the template components of a web page are of questionable value to the advertising system. The removal of template blocks should thus help the system to focus on the relevant content.

The general approach to extracting keywords from documents is to first consider each phrase or word in the target document as candidate. The importance of the candidate keyword to the document is evaluated based on various properties or features of the candidate, while the weights of those features are usually determined by machine learning algorithms such as Naive Bayes [6] or logistic regression [8,15]. No matter what learning algorithm is used, the quality of a keyword extraction system still heavily depends on the features.

Although many different attributes have been shown to be useful in this task, previous work has identified the three most important features capturing different properties of a candidate keyword: term frequency, document frequency and search query log. Term frequency provides a rough idea on how important this term is, relative to the target document. Document frequency downweights stopwords and common terms in the document collection. Finally, search query log provides an additional source of term importance.

5.1 Impact of Template-Blocks on Keyword Extraction

Intuitively, template blocks should be removed from the target page before a keyword extraction system is applied. After all, the extracted keywords are expected to be relevant to the content of the page, not the template blocks such as navigational side bars. In addition, terms that occur in one template block can often be found in several other template blocks on the same page. If template blocks are not removed before further processing, these terms will have higher term frequency and can be mistakenly judged as more relevant than terms that only appear in the real content block. These phenomena can be magnified if the terms happen to be frequently queried keywords. Nevertheless, good results were reported without first "de-noising" the documents by removing template blocks [15]. We hypothesize that the negative effect of template blocks is alleviated by the document frequency feature in their work. When the target document is similar to the document collection where document frequency is derived from, de-noising is implicitly done by the document frequency feature. In this case, terms that appear in the template block will not only have higher term frequency but also higher document frequency. As a result, the importance of these template terms will not be overly emphasized.

However, in practice, a machine learning based system is often faced with the problem of *domain adaptation* – the system is developed or trained in one

domain but deployed in another. Due to the different distribution of the data, performance degradation of the system is commonly observed. In this scenario, since the template in the training data may not appear in the testing data, document frequency derived using the training document collection can no longer downweight the importance of the new template terms, which may therefore be judged as relevant keywords.

5.2 Experiments

In order to validate the assumption that our template-block detection method is crucial to keyword extraction when faced with the cross-domain problem, we collected two sets of documents from different sources for experiments. The first set of documents, denoted as **IA**, is a random sample of 828 pages from the Internet Archive[2]. Since this dataset was used mainly for training the keyword extraction system, we intentionally made it more diversified. In particular, no two pages are originally from the same site. The second set of documents, denoted as **MSN**, is a collection of 477 pages from `MSN.com`, which covers topics such as *autos*, *health*, *finance* and etc. This dataset was used for testing the cross-domain behavior of the keyword extractor. Therefore, data set **IA** did not include any pages from **MSN.com**. Each page in these two datasets was annotated with relevant keywords for training and evaluation.

We followed the approach described in [15] to build a state-of-the-art web keyword extraction using dataset **IA**. We preserved all the features described in [15] except the linguistic features and the URL feature, which do not seem to change the performance much. Depending on the experiment setting, the document frequency may be derived from different collections of documents and our template-block detector may be used as a preprocessing step to remove template blocks. In the experiments, we used the template detector trained over the ODP data, using the random forest classifier and set to produce the expected template recall of 50%.

As suggested in [15], to measure the performance of keyword extraction we used the *top-N* score, calculated as follows. For a given document, we count how many of the top N keywords output by the system are "correct" (i.e., they were also picked by the annotator), and then divide this number by the maximum achievable number of correct predictions. Mathematically, if the set of top-N keywords output by the system is K_n and the set of keywords picked by the annotator for this document is A, then the *top-N* score for this test document is $|A \cap K_n| / \min(|A|, N)$. We report the average *Top-1,-2,-3,-5,-10* scores over the documents in the test set.

Table 1 presents the results of our first four different experimental settings, with datasets **IA** and **MSN** used for training and testing, respectively. The *ORD* configuration is the baseline, where the trained system is applied on the testing documents directly. The document frequency features used here were derived using dataset **IA** only. Assume that we do not know *a priori* where the test pages

[2] http://www.archive.org

Table 1. The results of keyword extraction: *ORD* – trained and tested as usual; *DN* – removing template-blocks in test pages before extracting keywords; *DF* – similar to *ORD* but the document frequency list is derived using both document sets **IA** and **MSN**; *DN-DF* – the combination of *DN* and *DF*.

	Top1	Top2	Top3	Top5	Top10
ORD	34.80	27.16	24.28	19.23	17.06
DN	37.74	28.11	25.48	22.14	20.55
DF	41.09	31.68	26.04	22.44	19.17
DN-DF	40.46	32.00	27.66	23.71	21.28

come from, i.e., we have no knowledge on which web site the keyword extractor will be deployed at training time. To reduce the negative influence brought by the template blocks, the *DN* (de-noising) configuration first applies our template detector to the *testing* pages to remove the template blocks. The same keyword extractor is then applied to the preprocessed pages. In Section 5.1, we assumed that if document frequency is derived using a document collection that is similar to the evaluation domain, then including the new document frequency features implicitly de-noises the data. This assumption is validated in the experimental setting *DF* (new document frequency). In this configuration, the document frequency statistic was derived using data sets **IA** and **MSN** together, and was used in both training and testing. Finally, the *DN-DF* configuration tests whether removing template-blocks in the testing documents can still enhance the results after using the new document frequency statistic.

From Table 1, we can clearly see that de-noising the data using our method does improves the results of keyword extraction. Comparing *DN* with *ORD*, this preprocessing step improves the *top-N* scores consistently (the differences are all statistically significant except for Top-2)[3]. As we can see, having a better document frequency list is still a more powerful solution *if the testing domain is known in advance*. The *top-N* scores of *DF* are consistently better than *ORD* and *DN* (statistically significant compared to all Top-*N* scores of *ORD*, but only statistically significant on Top-2 and Top-10 compared to *DN*). Interestingly, the result can be further improved with the preprocessing step of de-noising. Except the *Top-1* score, all other *top-N* scores of *DN-DF* are even better than *DF* (the differences on Top-3,-5,-10 are statistically significant).

In all the aforementioned experiments, de-noising was used only on the testing data but not on the training data. It is therefore interesting to examine whether using de-noised training data can further improve the results. Table 2 presents the results of this set of experiments. The *DN_both* configuration applies de-noising in both training and testing. *DN-DF_both* is similar to *DN_both* except that document frequency is derived using datasets **IA** and **MSN** together. Comparing these two results with the *DN* and *DN-DF* rows in Table 1, we can see that de-noising the training data does not seem to help. This may be due

[3] Statistical significance test was done using 2-tail paired-t test on the top-*N* scores of each test document. Significance was tested at the 95% level.

Table 2. The results of keyword extraction when (1) de-noising is applied on both training and testing data, and (2) a different document frequency static is used: *DN_both* – removing template-blocks in training and test pages before extracting keywords; *DN-DF_both* – similar to *DN_both* but the document frequency list is derived using both document sets **IA** and **MSN**; *ODP-DF* – when document frequency is derived using the pages in the ODP dataset (see text for details)

	Top1	Top2	Top3	Top5	Top10
DN_both	36.48	27.68	24.84	21.71	20.47
DN-DF_both	39.20	31.79	26.96	23.44	21.04
ODP-DF	29.35	25.58	23.57	19.14	16.82

to the fact that the document frequency derived from the in-domain document collection shadows the positive effect that de-noising can bring.

Finally, we want to emphasize that although using document frequency from the target domain documents has the implicit effect of de-noising, a DF table based on large collection of diversified documents cannot provide similar results. This is due to the fact that words in template blocks can no longer appear in most documents. To validate this point, we generated a DF table using the ODP dataset (see Section 4.2) and used it in both training and testing. As shown in the last row of Table 2, this result is the worst among all the settings we tested, probably because the distribution difference between the target domain (MSN dataset) and the ODP dataset.

6 Conclusions

We considered the problem of building a web-site independent template-block detection. Our contribution was to acknowledge the effectiveness of within-site template-block detection and use it to provide data for building cross-site template-block classifiers. The proposed methodology is accurate in identifying true template blocks as training examples. With the possible contamination of non-template data taken into account, our cross-site template detectors were able to achieve recall of as high as 70% at the precision of 90%. Given that the training and test data originated in different sites, this represents high level of accuracy of practical importance. We were also able to show that useful template detection models can be learnt with biased samples of individual websites. This further increases the flexibility of creating template detection models using readily available data.

Our methodology of template-block removal was assessed by its impact on the target application of keyword extraction. Template removal proved to be universally beneficial in this task, with relative increases in performance of as much as 20%, pointing at information extraction tasks as the class of applications where noise filtering is likely to improve performance.

Acknowledgement. We would like to thank David Gibson for sharing the dataset used in [7]. We are also grateful to Chris Meek for the helpful discussion on applying the work to keyword extraction.

References

1. Bar-Yossef, Z., Rajagopalan, S.: Template detection via data mining and its applications. In: Proc. of the 11th World Wide Web Conference (2002)
2. Cai, D., Yu, S., Wen, J., Ma, W.: VIPS: a vision-based page segmentation algorithm. Technical Report MSR-TR-2003-79, Microsoft Research Asia (2003)
3. Chen, L., Ye, S., Li, X.: Template detection for large scale search engines. In: Proceedings of the 21st Annual ACM Symposium on Applied Computing (SAC'06), pp. 1094–1098. ACM Press, New York (2006)
4. Debnath, S., Mitra, P., Pal, N., Giles, C.: Automatic identification of informative sections of web pages. IEEE Transactions on Knowledge and Data Engineering 17(9), 1233–1246 (2005)
5. Fayyad, U., Irani, K.: Multi-interval discretization of continuousvalued attributes for classification learning. In: Proceedings of the 13th International Joint Conference on Artificial Intelligence, pp. 1022–1029 (1993)
6. Frank, E., Paynter, G.W., Witten, I.H., Gutwin, C., Nevill-Manning, C.G.: Domain-specific keyphrase extraction. In: Proc. of IJCAI-99, pp. 668–673 (1999)
7. Gibson, D., Punera, K., Tomkins, A.: The volume and evolution of web page templates. In: Proc. of the 14th World Wide Web Conference, pp. 830–839 (2005)
8. Goodman, J., Carvalho, V.R.: Implicit queries for email. In: CEAS-05 (2005)
9. Kushmerick, N.: Learning to remove internet advertisements. In: Proceedings of AGENTS-99 (1999)
10. Song, R., Liu, H., Wen, J., Ma, W.: Learning block importance models for web pages. In: Proc. of the 13th World Wide Web Conference, pp. 203–211 (2004)
11. Turney, P.D.: Learning algorithms for keyphrase extraction. Information Retrieval 2(4), 303–336 (2000)
12. Turney, P.D.: Coherent keyphrase extraction via web mining. In: Proc. of IJCAI-03, pp. 434–439 (2003)
13. Yi, L., Liu, B.: Web page cleaning for web mining through feature weighting. In: Proc. of 18th International Joint Conference on Artificial Intelligence (2003)
14. Yi, L., Liu, B., Li, X.: Eliminating noisy information in web pages for data mining. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2003), ACM Press, New York (2003)
15. Yih, W., Goodman, J., Carvalho, V.: Finding advertising keywords on web pages. In: Proceedings of the 15th World Wide Web Conference (2006)