International Journal of Database Theory and Application
Vol.8, No.3 (2015), pp.311-322
http://dx.doi.org/10.14257/ijdta.2015.8.3.27

Web Data Extraction Based on Ensemble Learning

Yongquan Dong, Qiang Chu and Ping Ling

School of Computer Science and Technology, Jiangsu Normal University, Xuzhou 221116, China
tomdyq@163.com, 785469372@qq.com, lingicehan@163.cn

Abstract

With the rapid development of Internet technology, Web has become a huge information source with massive amounts of data. But these data are usually embedded in the semi-structured pages. In order to use these data effectively, the primary problem is to extract the data and store them in structured form. Most of current approaches use a single classifier to extract web data, but relying on a single classifier is not sufficient and different classifier has different performance for the same problem. In this paper, we use the method of ensemble learning for web data extraction. Firstly, we parse the page as a Dom tree, identify the main data regions, and construct feature sets of text nodes in the region. Secondly, we choose multiple kinds of base classifiers (SVM, KNN and Random Forest) to build classification models and then use the linear method to integrate results of each classification model. Finally, we combine integration results with heuristic rules to get the final extraction results. The experiment results show that our approach outperforms the baseline approaches and has a good robustness.

Keywords: *Web Data Extraction; Ensemble Learning; Data Integration*

1. Introduction

With the rapid development of Internet technology, the data on the Web has increased dramatically. And there is an urgent need for people to obtain the required information from the Web. However, there is too much disorganized information in the web pages, and the amount of the information we need is only a small portion, so it is very inconvenient for users to search on the Web. In order to solve the problem, many researchers have conducted researches on web data extraction [1], which can identify the information from unstructured or semi-structured Web pages, and turn it into a structured format.

In this paper, we focus on regularly structured data which are produced by computer programs following some fixed templates. But these data may come from different templates. How to extract the data from Web pages generated by many different templates is non-trivial. One possible solution is that we first distinguish web pages generated by different templates, and then build an extractor for each template. We say that this type of solution is template-dependent. However, accurately identifying web pages for each template is not a trivial task because even web pages from the same website may be generated by dozens of templates. Even if we can distinguish web pages, template-dependent methods are still impractical because the learning and maintenance of so many different extractors for different templates will require substantial efforts.

Recent work has shown that using template-independent approaches to extract the same type of data objects is feasible and promising [2-5]. Existing approaches usually use a single classifier to extract web data, but relying on a single classifier is not sufficient and different classifier has different performance for the same problem. In order to improve the extraction performance, in this paper we introduce an approach based on ensemble learning for web data extraction. Firstly, we identify main data regions

ISSN: 2005-4270 IJDTA

according to the page structure, then generate feature set of text nodes (We call leaf nodes of the DOM [6] tree of the Web page as text nodes). Secondly, we choose the base classifiers to build classification models with feature set, and then use the linear method to integrate classification results of each base classification model. Finally, we combine integration results with heuristic rules to get the final extraction results. Experimental results on 20 different structural sites prove the validity of our approach.

The rest of this paper is organized as follows. First, the related work is discussed in Section 2. Then, our approach is introduced in Section 3. In Section 4, the experimental setup and results are shown. Finally, the conclusion is presented in Section 5.

2. Related Work

Wrapper learning approaches like [7-8] are template-dependent. They take in some manually labeled web pages and learn some extraction rules (i.e. wrappers). Since the learned wrappers can only be used to extract data from similar pages, maintaining the wrappers as web sites change will require great efforts. Furthermore, in wrapper learning a user must provide explicit information about each template. So it will be expensive to train a system that extracts data from many websites as in our application. [9-13] are also template-dependent, but they do not need labeled training samples. They automatically produce wrappers from a collection of similar web pages.

Recent work in [2-5] has shown the feasibility and promise of template-independent web data extraction. MDR[2] is a wrapper induction system that does not require training. The main induction process of MDR is based on a string matching algorithm to calculate the string edit distance between each data object. DEPTA[3] is an improved version of MDR. It uses visual information to generate the DOM tree of web pages and uses a tree alignment algorithm for web data extraction But, [3] detects data objects only using tree regularities and not consider semantics. Furthermore, the data extracted by [3] have no semantic labels. Our approach uses rich features which not only contain structural features but also include content features. And the final results of our approach have semantic labels. [4] uses a clustering approach for automatic data extraction. It uses the similarities of the data format and the data content to group text tokens into clusters and obtains the final extraction result. The above approach only adapt to list pages, not to detail pages. Our approach is able to adapt to two kinds of pages. The work in [5] is similar to our approach. It treats web data extraction as a classification problem. It uses support vector machine to identify the start and end tags for a single attribute. Its problem is that it only uses a single classifier to extract the data, and the extraction accuracy still needs to be further improved. As different single classifier has different performance on web data extraction, so we use ensemble learning methods to extract web data.

3. Our Approach

3.1. Framework

The framework is divided two phrases. The first one is training phrase. In this phrase, we first identify the main data regions of training web sites to determine the area of data extraction (Section 3.2). Secondly, we choose multiple base classifiers to build models, and then learn the weights for each base classifier model on validation set (Section 3.3.1). The second one is extraction phrase. In this phrase, we first use every trained model on testing set to obtain every classification result. Secondly, we use the linear method to integrate these classification results, and then combine the integration results with heuristic rules to get the final extraction results (Section 3.3.2). The framework is shown in Figure 1.

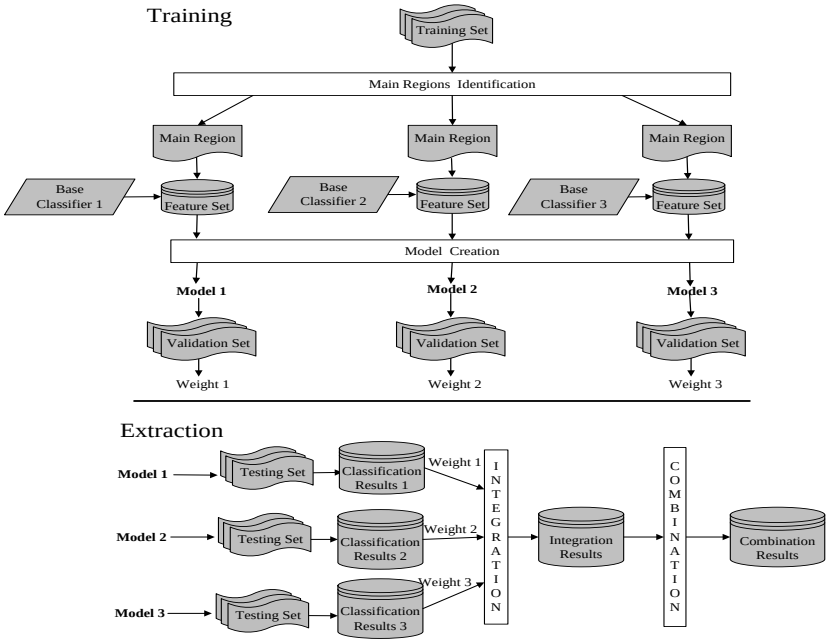

Figure 1. The Framework of Our Approach

3.2. Main Region Identification

A Web page contains a lot of data regions, but not all data regions are required by the user. Therefore, we must determine the data extraction region at first. In this paper, we define a region node as a non-leaf node of the DOM tree, and then set three parameters: linkCharacterNum, characterNum, linkCharacterDensity. The linkCharacterNum is used to record the number of hyperlink text characters of each region node. The characterNum is used to record the number of all text characters of each region node. The linkCharacterDensity is used to record the proportion of hyperlink text characters of each region node, which is shown in equation (1).linkCharacterDensity =linkCharacterNum/characterNum (1)

We have observed 100 Web pages from different websites and find that in the data region required by the user, its linkCharacterDensity is always less than 0.5. So in this paper we call the data region whose linkDensity is smaller than λ as the main region. According to our observation, we set λ to be 0.5. The main region is the concern of web data extraction system. As shown in Figure 2, the area covered by the red box represents the main region to be extracted.

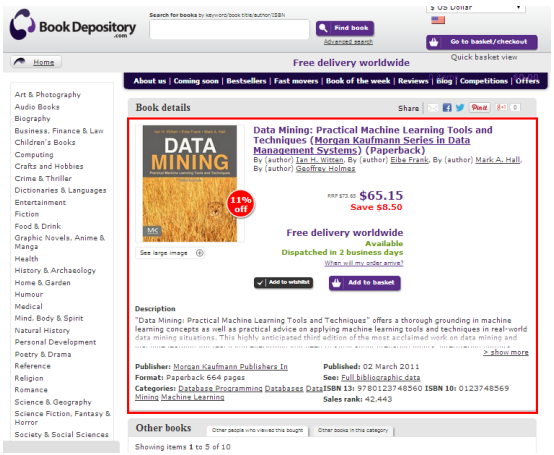

Figure 2. The Main Region of a Sampling Web Page

The algorithm to determine the main regions uses the idea of hierarchical traversal method, which is to traverse each region node of the html Dom tree, and judge the linkCharacterDensity of each one. When finding the linkCharacterDensity of one region node is less and equal to λ, the algorithm immediately stops the search for all region nodes below this region node and turns to retrieve other region nodes in other regions. The details are shown in Algorithm 1.

Algorithm 1: The algorithm to determine main regions

Input: web page p
Output: mainRegions

```
1.   rootOfT = parseHtmlToDomTree(p);  mainRegions={};
2.   InitQueue(E); EnQueue(E,rootOfT);
3.   while (IsEmpty(E)) do
4.     DeleteQueue(E, headOfQueue);
5.     linkCharacterNum=ComputeLinkCharacterNum(headOfQueue);
6.     characterNum=ComputeCharacterNum(headOfQueue);
7.     linkCharacterDensity=linkCharacterNum/characterNum;
8.     if(linkCharacterDensity < threshold)
9.       mainRegions.add(headOfQueue);
10.    else
11.      EnQueue(E, GetChildenRegionNodes(headOfQueue);
12.    endif
13.  Endwhile
14.  return mainRegions;
```

Algorithm 1 starts with parsing the web page p into a DOM tree whose root node is denoted as rootOfT and initializing the mainRegions which is used to store the region nodes of main regions. Line 2 initializes queue E which is used to keep the non-visited region nodes and pushes rootOfT into E. Lines 3-13 judge each region node to find main regions. Line 4 gets the head node of E headOfQueue which is current non-visited region node, then removes it. Lines 5-7 calculate the number of hyperlinks text characters, the number of all text characters and hyperlink density respectively. Lines 8-12 identify if the region node is main regions by hyperlink density. If its linkCharacterDensity is less than the threshold, the region node is added into mainRegions(Lines 8-9). Otherwise, its all children's region nodes are added into E (Lines 10-12). Finally, in line 14, the final main regions are returned by mainRegions.

3.3. Model Construction

In this section, we first introduce three base classifiers, then give the details of the integration approach of these base classifiers, and finally introduce the used feature set.

3.3.1. Base Classifier: In this paper, we select three base classifiers to create the classification model: SVM、KNN and Random Forest.

(1)SVM [14]

SVM is the abbreviation of Support Vector Machine which can be used for classification. It belongs to a family of generalized linear classification. SVM can simultaneously minimize the empirical classification error and maximize the geometric margin. SVM maps input vector to a higher dimensional space where a maximal separating hyperplane is constructed. Two parallel hyperplanes are constructed on each side of the hyperplane that separate the data. The separating hyperplane maximizes the distance between the two parallel hyperplanes.

(2)KNN [15]

KNN (k-Nearest Neighbor, KNN) is a classification algorithm based on distance measure. A sample is classified by a majority vote of its neighbors, with the sample being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). In this experiment, we set k=1, that is to say, the sample is assigned to the class of that single nearest neighbor.

(3)Random Forest [16]

Random Forest builds a forest containing a lot of decision trees randomly, and each decision tree has no correlation with each other. When a new sample is inputted, each decision tree in the forest will judge the sample, give their classification results and choose the classification having the most votes as the final classification result of the new sample.

3.3.2. Classifier Integration: In the previous section of this paper, we introduce three base classifiers. However, a single base classifier has its shortcomings in performance. In order to improve the accuracy of data extraction, we use ensemble learning [17][18] approach to integrate these base classifiers. The ensemble classifier determines the final result by combining the classification results of each base classifier. In this paper, we perform two integration methods: voting and linear integration.

(1)Voting

Voting[19] is a widely used integration method. At first, this method uses multiple base classifiers to get the individual classification result, and then utilizes the voting method to obtain the final classification result. The common voting methods include majority voting, one-vote veto, minimum voting, etc. Voting method is relatively simple, which uses each base classifier with the equal weight. But in reality, different base classifier has different performance for a problem. So we will take into account the weights of these base classifiers.

(2)Linear Integration

Let $c = \{c_1, c_2, ..., c_n\}$ represent a set of base classifiers, where c_i is the i-th base classifier, and the number of the base classifiers is n . Let $y = \{y_1, y_2, ..., y_k\}$ represent a set of categories, where y_i is the i-th category, and the number of the category is k . And x_{ij} represents the prediction probability of base classifier c_i for category y_j , and a_{ij} represents the weight of base classifier c_i for category y_j . The input is $X = \{X_1, X_2, ..., X_m\}$ representing feature sets of text nodes to be classified, where X_i is the feature set of i-th text node. Then the category of the text node whose feature set is X_i is computed with equation (2).

$$y_i = max[\sum_{i=1}^{n} a_{i1}x_{i1}, \sum_{i=1}^{n} a_{i2}x_{i2}, \sum_{i=1}^{n} a_{i3}x_{i3}, ..., \sum_{i=1}^{n} a_{ik}x_{ik}] \quad (2)$$

Here x_{ij} has been known, so we must find the solution of a_{ij} . We divide the data set into three parts: training set, validation set and testing set. The training set is used to train base classifiers. The validation set is used to obtain the weight of each base classifier for each category. And the testing set is used to get the extraction performance. Let $N = \{N_1, N_2, ..., N_k\}$, where N_i is the number of text nodes which belong to y_i in the validation set. And M_{ij} represents the number of the text nodes which belong to y_j in the validation set and is classified into y_j by base classifier c_i .Then the a_{ij} is computed with equation (3).

$$a_{ij} = \frac{M_{ij}}{N_j}, i = 1, 2, ..., \mathrm{n}; \mathrm{j} = 1, 2, ..., \mathrm{k} \quad (3)$$

3.3.3.Result Combination: The output categories of each attribute contain three types: the first one is the attribute name (denoted as [attribute]-an), the second one is the attribute value (denoted as [attribute]-av) and the third one is the attribute value and name (denoted as [attribute]-nv), where [attribute] represents any attribute. For example, the attribute of "author", It may have two situations: one is that the attribute name and attribute value of "author" are separated into the two text nodes, such as text nodes "author: " and "Ian H. Witten", and the categories of these two text nodes are "author-an" and "author-av" respectively; the other one is that the attribute name and attribute value of "author" are in one text node, such as text node "author: Ian H. Witten", and the category of this text node is "author-nv". For the first situation, we can directly combine the values with the same catetory "[attribute]-av" as the final extraction result of this attribute. For the second situation, we first separate its value into attribute name and attribute value according to some heuristic characteristics, then combine this value with the adjacent values having the same category "[attribute]-av" and regard the combination value as this attribute's final extraction result. The details are shown in Algorithm 2.

Algorithm 2: The algorithm to combine classification results

Input: C and als, where C is the set of classification results and als is the set of attribute labels. Each element of C contains two fields: "content" and "category". The field "content" denotes the content of a text node and the field "category" denotes the attribute category which may be [attribute]-an, [attribute]-av or [attribute]-nv.

Output: R, the set of extraction result

```
1.  R = {}; S = ''; i=1;
2.  while i <= length(C) do
3.    for j = 1 to length(als) do
4.      if C[i].category == als[j]+'-av' then
5.        S=S+C[i].content;
6.        while(C[++i].category== als[j]+'-av')
7.          S=S+C[i].content;
8.        end while
9.        R.put(als[j], S); S=''; continue;
10.     endif
11.     if C[i].category == als[j]+'-nv' then
12.       q = separateNV(C[i].content);
13.       S=S+getAV(C[i].content, q);
14.       while(C[++i].category== als[j]+'-av');
15.         S=S+C[i].content;
16.       end while
17.       R.put(als[j],S);  S='';  continue;
18.     endif
19.     i = i + 1;
20.   end for
21.  end while
22. return R;
```

Algorithm 2 starts with initializing the extraction result R and S which is used to store the value of each attribute. Lines 2-21 judge each classification result to get the final extraction result R. In lines 4-10, if the category of the classification result is "av" of an attribute, we combine all contents with the same "av" of this attribute into a group and put the attribute name and its value into R. In lines 11-18, if the category of the classification result is "nv" of an attribute, we first separate it into attribute name and attribute value by some specific symbols such as colon, then we combine all contents with the same "av" of this attribute into a group judges the attribute and put the attribute name and its value into R. Finally, in line 22, the final result is returned by R.

3.3.3. Feature Set: In this paper, we use three kinds of features for a text node:

(1)Own features. They include whether it contains numbers, whether it contains special characters (for example, ¥, :) and the length of text node.

(2)Path features. They include the path information of text node.

(3)Contextual features. They include the information of the adjacent previous text node and the adjacent next text node. And this information is composed of the own features of the text node.

4. Experiments

4.1. Dataset

In this paper, the training set is collected from 20 websites with different structures. On each website we collect 40 pages. The validation set and the testing set is also collected from the same 20 websites. On each website we collect 10 pages.

4.2. Category

We extract 21 kinds of attributes which contain “title”, “author”, “market price”, “website price”, “discount”, “publisher”, “publication date”, “edition number”, “page number”, “word number”, “printing date”, “format”, “paper size”, “printing number”, “ISBN”, “package”, “language”, “size”, “weight”, “ASIN”, “member price”. Each attribute contains three types of category: “[attribute]-an” represents the name of the attribute; “[attribute]-av” represents the value of the attribute; “[attribute]-nv” represents the name and value of the attribute, where “[attribute]” can be replaced with any kind of the above attributes. For example, in text node “author:”, its category is “author-an”; in text node “Ian H. Witten”, its category is “author-av”; in text node “author: Ian H. Witten”, its category is “author-nv”.

4.3. Evaluation Criteria

We measure the extraction performance via three metrics: precision, recall and F1 [20]. A brief definition is given as follows.

Defining m as the existing total number of attributes, t as the number of attributes which are correctly extracted, n as the number of attributes which are wrongly labeled. Then the evaluation criteria is defined as follows: $P=t/(t+n)$, $R=t/m$, $F1=2PR/(P+R)$,where P is precision representing the confidence of the results, R is recall representing the coverage of correct results, F1 is the result of considering P with R together. As every attribute has its F1, for simplicity, we only use average F1 of all attributes to evaluate the experiment in this paper.

4.4. Discussion on Experimental Results

We have designed four experiments to evaluate the effects of our approach. In these experiments, the base classifier Random Forest is abbreviated as RF, and the ensemble classifier integrated by SVM, KNN and Random Forest is abbreviated as SKR.

(1)Performance Comparison between Single Base Classifier and Ensemble Classifier

In this experiment, we compare the extraction performance of single base classifier with ensemble classifier. The result is shown in Figure 3.

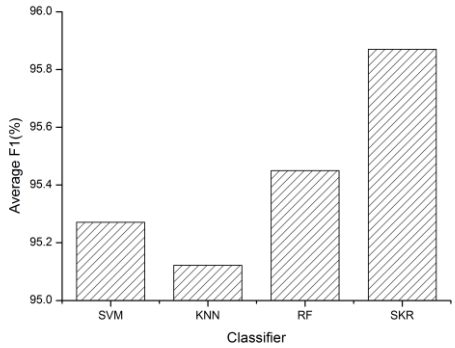

Figure 3. Average F1 Using Single base Classifiers and Ensemble Classifier

In Figure 3, the average F1 of single base classifiers SVM、KNN and Random Forest are 95.27%, 95.12% and 95.45% respectively. The average F1 of ensemble classifier is 95.87%. Compared with single base classifiers, ensemble classifier improves the average F1 by 0.42%-0.75%. The result suggests the extraction performance of ensemble classifier outperforms that of any single base classifier. This is because although each single base classifier has its own advantages and disadvantages, ensemble classifier which integrates multiple base classifiers can utilize complementary information and achieve the best performance. In the following experiments, we use ensemble classifier for Web data extraction.

(2) Effect of Changing the Size of Training Set

In this experiment, we test the influence of training set size on the extraction results. There are 20 websites with different structures in training set. From each website, we randomly select the pages whose size is from 5 to 40 with step size 5 to derive different training sets. The experimental result is shown in Figure 4.

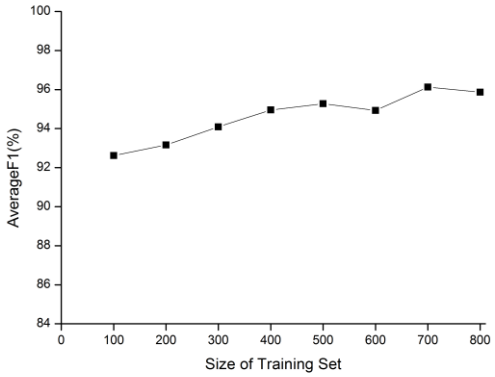

Figure 4. Average F1 using Different Size of Training Sets

In Figure 4, we can see that when increasing the size of the training set, a gradual improvement in average F1 is obtained. It can also be seen from Fig. 4, when the size of training set is 500, the average F1 is close to its maximum. Thus, our method only needs labelling a few pages to achieve better extraction performance and it can be applied to large-scale Web data extraction.

(3) Performance Comparison between Voting Integration and Linear Integration

In this experiment, we compare the extraction performance of voting integration and linear integration. In this experiment, the voting integration and linear

integration use the same training set, test set and the base classifiers. The experimental result is shown in Figure5.

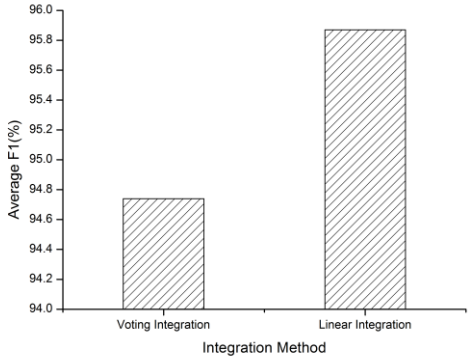

Figure 5. Average F1 using Voting Integration and Linear Integration

Figure 5 shows that the average F1 of the ensemble classifier integrated by linear method reaches 95.87%, while that integrated by voting method is 94.74%. Therefore the linear integration is better than voting integration. This is because that different base classifier for the extraction problem has different performance.

(4) Robustness of Ensemble Classifier

In this experiment, we test the robustness of the ensemble classifier. The testing set of this experiment is 20 pages collected from a new website which does not exist in the 20 websites of training set. The experimental result is shown in Table 1.

Table 1. Robustness of Ensemble Classifier

Average R(%)	Average P(%)	Average F1(%)
74.29	98.11	84.55

From Table 1, we can see that the value of average F1 is 84.55%. Thus, the ensemble classifier has strong robustness for the new website.

5. Conclusion

In this paper, we use the method of ensemble learning to extract Web data. Firstly, we clean the Web page, and build the feature set of text nodes in the main regions. Secondly, we choose three base classifiers and use feature set to construct classification models respectively. Then, we use linear method to integrate classification results of each base classifier model. Finally, we use heuristic method to combine integration result to get the final extraction results. The experiments prove that the extraction performance of our approach is better than that of each base classifier. And our approach has a good robustness.

The text nodes in this paper all contain one attribute. However, the structure of Webpage is complex. Many text nodes contain multiple attributes. Therefore, we will further study the problems that the single text node contains multiple attributes in the future work.

Acknowledgements

This work is supported by the National Natural Science Foundation of China under Grant No. 61100167 and 61105129, the Natural Science Foundation of Jiangsu Province, China under Grant No. BK2011204, Qing Lan Project, the National Training Program of

Innovation and Entrepreneurship for Undergraduates under Grant No. 201310320052, the Practice Innovation Training Program Project for the Jiangsu College Students under Grant No. 201310320052Z and the Graduate Scientific Research and Innovation Project of Jiangsu Normal University under Grant No. 2013YYB130.

References

[1] C. H. Chang, "A Survey of Web Information Extraction Systems", IEEE Transactions on Knowledge and Data Engineering, vol. 18, no. 10, **(2006)**, pp. 1411-1428.
[2] B. Liu, R. Grossman, Y. Zhai, "Mining data records in web pages", Proceedings of the 9th ACM international conference on Knowledge discovery and data mining(SIGKDD), **(2003)**, pp. 601-606.
[3] Y. Zhai, B. Liu. "Web data extraction based on partial tree alignment", Proceedings of the 14th international conference on World Wide Web, **(2005)**, pp. 76-85.
[4] X. Gao, L. P. B. Vuong, M. Zhang, "Detecting data records in semi-structured web sites based on text token clustering", Integrated Computer-Aided Engineering, vol. 15, no. 4, **(2008)**, pp. 297-311.
[5] A. Finn, N. Kushmerick, "Multi-level boundary classification for information extraction", Proceedings of the 15th European Conference on Machine Learning(ECML), **(2004)**, pp. 111-122.
[6] World Wide Web Consortium. W3C, "The Document Object Model", http://www.w3.org/DOM, **(2014)**.
[7] S. Zheng, R. Song, J. Wen, C. L. Giles, "Efficient record-level wrapper induction", Proceedings of the 18th ACM Conference on Information and knowledge management(CIKM), **(2009)**, pp. 47-56.
[8] N. Kushmerick, "Wrapper induction: efficiency and expressiveness", Artificial Intelligence, **(2000)**, vol.118, no. 1-2, pp. 15-68.
[9] T. Grigalis, "Towards web-scale structured web data extraction", Proceedings of the 6th ACM international conference on Web Search and Data Mining(WSDM), **(2013)**, pp. 753-758.
[10] H. Zhao, W. Meng, Z. Wu, V. Raghavan, C. Yu, "Fully Automatic Wrapper Generation for Search Engines", Proceedings of the 14th international conference on World Wide Web(WWW), **(2005)**.
[11] C. H. Chang, S. Lui, "IEPAD: Information Extraction Based on Pattern Discovery", Proceeding of the 10th international conference on World Wide Web(WWW), **(2001)**, pp. 681-688.
[12] C. Valter, M. Giansalvatore, M. Paolo. "ROADRUNNER: Towards Automatic Data Extraction from Large Web Sites", Proceedings of the 27th International Conference on Very Large Data Bases(VLDB), **(2001)**, pp. 109-118.
[13] A. Arasu, H. G. Molina, "Extracting Structured Data from Web Pages", Proceedings of the ACM International Conference on Management of data(SIGMOD), **(2003)**, pp. 337-348.
[14] V. Vapnik, "The Nature of Statistical Learning Theory", NY: Springer-Verlag, **(1995)**.
[15] T. Cover, P. Hart, "Nearest neighbor pattern classification", IEEE Transaction on Information Theory, vol. 13, no. 1, **(1967)**, pp. 21-27.
[16] B. Leo, "Random Forest", Machine Learning, vol. 45, no. 1, **(2001)**, pp. 5-32.
[17] B. Lazzerini, S. L. Volpi, "Classifier ensembles to improve the robustness to noise of bearing fault diagnosis", Pattern Analysis and Applications, vol. 16, no. 2, **(2003)**, pp. 235-251.
[18] G.I. Webb, "Multistrategy ensemble learning: reducing error by combining ensemble learning techniques", IEEE Transactions on Knowledge and Data Engineering, **(2004)**, vol. 16, no. 8, pp. 980-991.
[19] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee, "Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods. Annals of Statistics", vol. 26, no. 5, **(1998)**, pp.1651-1686.
[20] C. J. Van Rijsbergen. Information Retrieval. Butterworths, **(1979)**.

Authors

Yongquan Dong, he is an associate professor at School of Computer Science and Technology, Jiangsu Normal University. He received his Ph.D. degree from School of Computer Science and Technology, Shandong University. His main research interests include Web information integration and Web data management.

Qiang Chu is a postgraduate student at School of Computer Science and Technology, Jiangsu Normal University since 2012. His main research interests include Web information integration.

Ping Ling is an associate professor at School of Computer Science and Technology, Jiangsu Normal University. He received his Ph.D. degree from School of Computer Science and Technology, Jilin University. Her main research interests include data mining.