Simultaneous product attribute name and value extraction from web pages

Bo Wu, Xueqi Cheng, Yu Wang, Yan Guo, Linhai Song
Institute of computing technology
Chinese academy of sciences
Beijing, China
bowu365@gmail.com, cxq@ict.ac.cn, wangyu05@software.ict.ac.cn,guoy@ict.ac.cn, songlinhai@software.ict.ac.cn

***Abstract*—Much work has been done in the area of template-independent web data extraction. However, these approaches deal with the attribute value extraction and annotation either in separate phases or constrained to a predefined set of attributes which is highly ineffective. In this paper, we perform the attribute extraction and annotation simultaneously by extracting the attribute name and value pair at the same time. In our approach, we use a co-training algorithm with naive Bayesian classifier to identify the candidate attribute name and value pairs in the unlabeled pages. The candidate attribute name and value pairs are used to detect the specification block of the product in web pages. Finally, all the attribute name and value pairs in the specification block are discovered. We conduct experiments for three types of products and obtain a promising result.**

***Keywords*-information extraction; web mining; template-independent; semi-supervised**

I. INTRODUCTION

The World Wide Web (www) contains a huge number of online shops which sell various kinds of products. The product information is embedded in web pages of different websites which imposes barrier on collecting, comparing and analyzing these product information universally. Thus, services which provide a structured search results such as some vertical search engine and comparative shopping begin to come forth. These services require to collect a given product information from various web pages of different websites in a template-independent manner.

The task of extracting information from web pages usually can be categorized into template-dependent or template-independent according to whether the pages needed to be extracted belong to the same template. For template-dependent extraction methods, as the product information is embedded in many websites of different templates, preparing training data site by site is tedious and a heavy burden [1] and the unsupervised template-dependent methods[2] cannot normalize the extracted fields of different websites. Thus the template-independent method is preferred. Recent work has shown that using template-independent approaches to extract meta-data from the same type of real-world object is feasible and promising. However, existing methods either attempt to do attribute extraction and annotation in two separate phases or only extract some pre-defined attributes of a object [3][4], which often lead to imprecise semantic annotation of the extracted date and error accumulation. We illustrate our motivation with an example. Fig 1 is a sample web page of a book which presents the detail information of the book. We will refer this kind of web page as *detail page* and the block containing product attributes information as *specification block*. To make the information easy to understand for human browsers, these product detail pages often present the attribute name and value together. The attribute name acts as the textual description of the value without which it is hard for human reader to understand the meaning of the value. For instance, if we see 10 digits without the ”ISBN” before it in Fig 1, it is hard to figure out what dose the 10 numbers refer to. However, existing methods fails to take advantage of the co-occurrence of the attribute name and value. we refer to such attribute name and value pair as *NVP* at the rest of the paper. [4] first extracts the product attribute values for the web pages and then asks user to annotate the clustered values of the same attribute. It would accumulate the error of the two phases. In addition, it is hard to predict the exact attribute just from the attribute value in many occasions, such as the ”author”and ”Narrator” in Fig 1. [3] gather the attribute value of a pre-defined set of attributes and cannot deal with the unseen attributes.

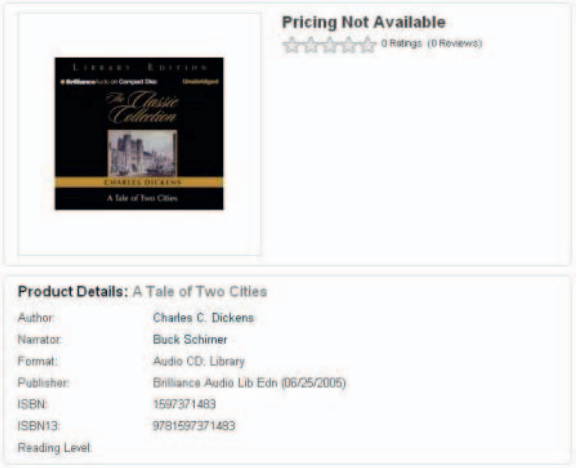

Figure 1. a sample page of a book

In this paper, we propose a semi-supervised template-

978-0-7695-3801-3/09 $26.00 © 2009 IEEE
DOI 10.1109/WI-IAT.2009.286

independent method that simultaneously extracts product attribute name and value from web pages for a given object. The web page is parsed into DOM tree and then both the value and name extraction can be considered as the task of classifying nodes in the DOM tree. Firstly, we use a co-training algorithm with naive Bayesian classifiers to identify the candidate NVPs in the page. Then the specification block is identified according to the sum of the confidence of all the candiate NVPs in a block. Finally all the NVPs in the specification block are identified using the layout, format and tree alignment information. In contrast to existing method, our method leverages the results of attribute name identification for extracting attribute value and at the same time benefit from the detected attribute value to discover attribute name. By incorporation of this co-occurrence of the attribute name and value, our method not only gets the exact semantic meaning of the extracted value but also acquire a more accurate result.

II. Overview of our system

Our system consists of 4 modules:

1) **Data collection**:We develop a web crawler which can automatically compose the query for a given object and feed the query to google to obtain the pages which contain the specification of the given product. The crawler then gathers the web pages using the returned URLs in the search results. The gathered web pages are then filtered using some heuristic rules to acquire the pages which contain the specification of the product.
2) **Seed generation**:We randomly choose several crawled pages and label the NVPs in them.
3) **Candidate NVP identification**:The name and value classifiers are trained on the labeled web pages to discover the candidate NVPs in the unlabeled pages. Every candidate NVP is associated with a confidence generated by the classifier.
4) **Specification Block detection**:With the candidate NVPs and their confidence, we can identify the specification block in the page and all the NVPs in this block with some heuristic rules. The final identified NVPs can then be added into the training set to enhance the classifiers.

In the section below, we will illustrate seed generation, the candidate NVP identification and specification block detection in details. Because the design of crawler is another widely studied area, we would not illustrate in detail here.

III. Seed generation

We randomly choose several crawled pages and then label the NVPs in these pages. The labeled pages are parsed into DOM trees. Firstly, the texts of all text nodes in the tree are collected and cleaned by removing stopwords and meaningless symbols. Secondly, the cleaned texts are stemmed and the number in the text is replaced with #NUM. Finally, the processed texts are saved as the training data. In the training data, the text nodes corresponding to the attribute name or value server as the positive examples and the rest text nodes server as the negative examples in training.

IV. Candidate NVP detection

Since the NVP consists of the attribute name and value, the co-occurrence of name and value provides us redundant features compared with only using name or value to identify the NVPs in the page. To make use of the redundancy in the features, we use co-training [5] algorithm which is semi-supervised algorithm that can combines the labeled data and unlabeled data together to reduce the required amount of labeled data for training accurate classifier. In our co-training setting, every NVP can be expressed using the attribute name in one view and can be expressed by the attribute value in the other view. We train a name and value classifier on the training data acquired on the seed generation step. The name classifier categorizes the text node in two classes: *name* or *negName* and the value classifier also categorizes the text nodes in two classes: *value* or *negValue*. Since the attribute name and value for a particular type of product usually consist of limited keywords, it is possible to identify the candidate attribute name and value in the page using the content feature of the text nodes. We use naive Bayesian classifier as the name and value classifier. [6] shows that parameter estimation in naive Bayesian classifier based on multinomial distribution is more affected by long documents than by short documents. Since the attribute name and value are often short texts which may be greatly affected, to capture the occurrences of the word and eliminate the bias for long documents, our naive Bayesian classifier is based on the multinomial distribution and the occurrence of the word is normalized by the document length. Moreover, as the attribute name and value are contiguous and aligned horizontally or vertically, the identified candidate NVP must conform to this layout or else be discarded.

To discover the candidate NVP in the page, we iterative run the name and value classifiers for a number of iterations or until no new NVP is identified. When the name or value classifier has recognized one text node as positive and the confidence of the classifier is above the threshold, it tries to find the contiguous text node. If the contiguous text node is aligned horizontally or vertically with the identified name or value node and the name node is before the value node or above the value node, we add this pair of node to the list of candidate NVPs. The name node and value node in the NVP are added as positive examples into the training data. While the classifier has recognized one text node as negative node and the confidence is above the threshold, the text node is added as negative example into the training data. The two

classifiers are then trained on the original labeled training data plus the newly added training data in the next iteration.

We have to point out that the newly identified training data are temporarily added into the training data. Because co-training algorithm is semi-supervised algorithm which uses the training data generate by the two views of itself, the incorrectly classified text nodes would have a great bad influence on the result of the classifiers in a long run. In our approach, the newly added training data during the co-training algorithm proceeding are removed from the training data when the co-training algorithm terminates.

V. Specification block detection and NVP identification

Base on the observation of the NVP and product specification in the page, we make the hypothesizes below to identify the specification block and NVPs in the page.

1) **Hypothesis I**:NVPs for a given object are supposed to appear in a specification block which corresponds to one node in the DOM tree. The NVPs which share a similar tree structure with each other are contained in the subtrees of specification block.
2) **Hypothesis II**:The name node and value node of one NVP are contiguous text nodes and aligned vertically or horizontally. The name node is before or above the value node.
3) **Hypothesis III**: The name nodes in a specification block share the same format so do the value nodes in the specification block.

With the generated candidate NVPs and their confidence which is a probability produced by the classifier, the specification block can be recognized. The final NVPs in the specification block then can be extracted. Firstly, we cluster the candidate NVPs according to the block which they belong to and add the confidence of all the candidate NVPs in the block. The block which has the biggest sum confidence is chosen as the specification block. Secondly, when a specification block is identified, we search the sibling node in the tree to find whether its sibling nodes have similar tree structure. If the specification block has several siblings which share the similar tree structure with it, this page is discarded as it would contain several specification blocks of different products. Thirdly, if there is only one specification block in the page, we find the all the subtrees which have the similar tree structure with the identified candidate NVP in the block. In these subtrees, the text nodes which share the same format information with the candidate NVP are treated as NVPs. Finally, the NVPs extracted from the specification block are treaded as the final NVPs of the product. These NVPs are labeled in the unlabeled page and then this page is added as training page.

Table I
TOP TEN ATTRIBUTE ACQUIRED BY OUR SYSTEM

Category	Top ten acquired attribute name
Digital camera	lens system, record media, dimension, pixels, aspect ratio, battery, image type, image processor, manually focus, optic zoom
laptop	manufacture warranty, max battery life, mobile technology, price, display type, processor speed, hard driver capacity, processor brand, operating system, weight
book	list price, seller, price, product dimension, ship weight, author, language, page, publisher, isbn

Table II
THE PRECISION, RECALL AND F1 FOR ALL THE PAGES IN THREE CATEGORIES

		Digital camera	Laptop	Book
precision	name_c	0.9056	0.8347	0.8754
	value_c	0.5127	0.5332	0.6045
	NVP_c	0.9173	0.9213	0.9425
recall	name_c	0.9127	0.7453	0.8354
	value_c	0.4732	0.4937	0.5413
	NVP_c	0.9255	0.8417	0.9167
F1	name_c	0.9091	0.7874	0.8549
	value_c	0.4922	0.6216	0.5711
	NVP_c	0.9214	0.8797	0.9294

VI. Experiment

In this section, we report the experimental results of our NVP extraction approach. With the crawler described above, we gather web pages of three types of products: book, notebook, digital camera. Every category contains 12 different objects and each object contains about 10 detail pages from different websites. We randomly choose three pages in a category from all the pages in the category to label the NVPs. Table I shows the top 10 attributes of each category discovered by our system which are ranked by the number of occurrence. By obtaining the attribute name of each attribute, we can get accurate semantic meaning of the extracted value, such as the list price and price in the book category, etc.

The extracted results are compared with the manually annotated answers. Table II shows the precise, recall and F1 of using attribute name only, using attribute value and using the attribute name and value pair in the three categories. name_c denotes that we only use name classifier to detect the candidate NVPs in the page. value_c denotes that we only use value classifier to detect the candidate NVPs in the page. We use self-training algorithm instead of co-training algorithm to learn the classifiers. The most confident unlabeled nodes with their predicted labels are added to the training data in the self-training algorithm. The NVP_c denotes our approach.

For a particular product in a category, we not only need to know the precision and recall of the extracted NVP from web pages. We also want to know the proportion of the extracted attributes to the attributes which the product owns. We refer

this measurement as coverage. Since some detail product pages may violate hypothesizes which we make in the above section, the NVPs in these pages cannot be extracted properly. Thus, the recall would suffer a decrease. However, as the internet contains many web pages which describe the same product, we can simply increase the number of the gathered pages to acquire the specification of the product. In our experiment, every category contains 12 products and every product contains about 10 pages which describe the specification of it. In this experiment, we manually collect the attributes of the products of the three categories which need to be extracted. Our approach achieves nearly 100% coverage of the attribute which need to be extracted.

VII. Related work

Our work is in the area of the web information extraction.

Wrapper learning methods like [1] are template-dependent. They take in some manually labeled web pages and then learn some extraction rules to extract the target date in the unlabeled pages. Since it already requires the user to label the target date in the training pages, it can get the exact semantic meaning of the extracted data. However, this kind of methods can only extract the web pages which belong to the same template. Maintaining wrappers for different websites is heavy burden. Furthermore, the wrappers have to be regenerated while the websites change.

[2] automatically extract the value from the web pages through comparing several pages which belong to the same template. However, due to the automatic nature of the approach, the data extracted have anonymous names. Thus, [7] presents an approach to get the semantic meaning of the extracted data. The names of the extracted data are assumed to be a part of the common template. With a set of heuristic rules it can establish the correct association of the extracted data and the name. However, this approach is template-dependent which would suffer the updating of the website and heavy burden of maintain the wrappers for each website. In addition, extracting value and name in separate phases leads to error accumulation.

[8] presents template-independent method which aims to acquire the attribute-value pair (AVP) of a given object. It consists of two phases. It firstly gathers the attribute names for a given class. Secondly, using the gathered attribute and a set of heuristic rules, it identifies the pages and block which would contain AVP and extract AVP from the web pages. The quality of the gathered attribute names would have a great influence extraction result. In their approach they design some rules to find the candidate attribute names and then filter the erroneous acquired attribute names using the hit counts and site frequency of search engine. Obviously, these attribute names need to be checked by human beings. In addition, the same attribute name in different websites would have various forms. Using the exact name of the attribute to extract the values would lead to low recall.

[3] proposes an integrated model based on hierarchical CRF and semi-CRF for detecting records and extracting attributes from raw web pages which is a template-independent extraction method. However, the attributes to be extracted are pre-defined and hence it cannot discover unseen attributes. [4] proposes a unsupervised template-independent extraction method. Their approach is based on a probabilistic graphical model to extract and normalize attributes from multiple websites at the same time. However, their approach require user to annotate the normalize attribute value. This decouple strategy would lead to error accumulation. Moreover, in many occasions, it is hard to predicate the attribute only with the normalized attribute value.

VIII. Conclusion

In this paper, we present a novel template-independent approach to extract product attribute name and value. The approach can not only get the exact semantic meaning of the extracted value but also acquire a more accurate result. We start with few training pages for a domain and then use a semi-supervised method to identify all candidate NVPs in the unlabeled pages. By detecting the specification block of the page, the final NVPs are identified with some hypothesizes. Our experiments show that our method has obtain a promising result, and using co-occurrences of attribute name and value can improve the result for both value and name extraction.

References

[1] I. Muslea, S. Minton, and C. Knoblock, "A hierarchical approach to wrapper induction," pp. 190–197, 1999.

[2] V. Crescenzi, G. Mecca, and P. Merialdo, "Roadrunner: automatic data extraction from data-intensive web sites," pp. 624–624, 2002.

[3] J. Zhu, Z. Nie, J.-R. Wen, B. Zhang, and W.-Y. Ma, "Simultaneous record detection and attribute labeling in web data extraction," pp. 494–503, 2006.

[4] T.-L. Wong, W. Lam, and T.-S. Wong, "An unsupervised framework for extracting and normalizing product attributes from multiple web sites," pp. 35–42, 2008.

[5] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," pp. 92–100, 1998.

[6] S.-B. Kim, K.-S. Han, H.-C. Rim, and S. H. Myaeng, "Some effective techniques for naive bayes text classification," *IEEE Trans. on Knowl. and Data Eng.*, vol. 18, no. 11, pp. 1457–1466, 2006.

[7] L. Arlotta, V. Crescenzi, G. Mecca, P. Merialdo, and U. R. Tre, "Automatic annotation of data extracted from large web sites," pp. 7–12, 2003.

[8] N. Yoshinaga and K. Torisawa, "Open-domain attribute-value acquisition from semi-structured texts."