A Dynamic Learning Framework to Thoroughly Extract Structured Data from Web Pages without Human Efforts

Dandan Song, Yunpeng Wu, Lejian Liao; Long Li, Fei Sun
Beijing Engineering Research Center of High Volume language Information Processing & Cloud Computing
Beijing Lab of Intelligent Information Technology
School of Computer Science, Beijing Institute of Technology
Beijing, China
sdd@bit.edu.cn, yunpengwu88@gmail.com, {liao1j, 2120101153, ofey}@bit.edu.cn

ABSTRACT
Tremendous concrete and comprehensive information is contained in structured data of web pages. Attributes and their corresponding values of entities are precious resources for automatic semantic annotation, knowledge discovery, and information utilization. However, various displaying styles and formats of web pages make it a challenging task to extract them. Based on our observation, despite the lack of information in a single page, different web pages and different web sites illustrating similar entities can provide adequate knowledge for computers to learn. This paper presents a dynamic learning framework to effectively extract structured information from enormous websites in various verticals (e.g., books, cameras, jobs). Different with other existing approaches that are static, require manually labeling samples and can not be flexible to unseen attributes, our approach aims at dynamically, automatically and thoroughly extracting structured data from web pages. Experiments with totally 17,850 web pages in 4 verticals demonstrated the effectiveness of our framework.

Categories and Subject Descriptors
H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval - Metadata; D.3.1 [Information Storage and Retrieval]: Information Search and Retrieval - Retrieval Models

Keywords
information extraction, structured data, learning framework

1. INTRODUCTION
With explosive growth of the internet, huge amounts of information is presented by structured data, which are generated from background databases, and displayed in a semi-structured form of attributes and their associated values for an entity. For example, for a web page illustrating a book, as shown in Figure 1, the attributes (title, author, publisher and etc.) and their corresponding values of the book are commonly listed. These online structured data make the internet a big pool of information, which provides new opportunities for revolution of some crucial applications, including automatic semantic annotations, knowledge database establishments and further information utilizations.

There are many existing methods for structured data extraction from web pages. They are manually tailored wrappers [1, 12] in early days, and later wrapper learning techniques [14, 18]. Complex generative models [19, 20], bootstrapping [13], interactive learning [9, 10] and probabilistic graphical model [16] are also developed. There are also unsupervised template detection approaches, such as RoadRunner [4] and EXALG [2]. Recently, a unified solution [8] and a Bayesian approach [15] are proposed.

Along with the method development, human effort is firstly exhaustively required and later gradually reduced, with much attention paid. However, manual work on semantically marking discovery results, preparing training pages, or labeling attributes is still required. Accordingly, these methods are constrained by predetermined attributes, thus are not flexible to unseen attributes. Furthermore, they are offline static systems, as once the model is learned, they are fixed and will be out in contrast to constant change of the internet.

On the other hand, as computers have strong abilities to learn knowledge, dynamically online learning techniques have successfully innovated multiple applications. For example, artificial intelligent system “Watson” won the quiz show Jeopardy [7], and the Never-Ending Language Learning (NELL) extracts knowledge and entity relationships [3].

In this paper, we aim at dynamically, automatically and thoroughly extracting structured data from multiple web sites of various verticals without human efforts. To our observation, based on a great number of web resources, there is adequate information with similar layout structures, on different web pages, and among different web sites that can be learned by computers for structured data extraction. Therefore, crucial training samples in traditional supervised wrapper learning methods can be automatically attained, strictly filtered and dynamically updated by computers themselves, without any human efforts. Based on this knowledge specif-
Correspondingly, our framework is composed of two procedures – a Credible Attributes Learning System, and an Items Discovery and Extraction System (In the rest of the paper, we treat an attribute-value pair as a whole, which will be referred to as an Item). In the first procedure for learning credible attributes, by using structural features, inner-site features, and cross-sites features of web pages, an automatic attribute learning system is designed to gather potential items, to get candidate attributes, and then to generate credible attributes. Importantly, it is a dynamic online learning system, and thus the credible attribute results will be more accurate and reliable, with the increasing of input web pages. Then based on these credible attributes, the following items discovery and extraction procedure is implemented to detect item discovery fields, and to discover and extract existing credible and new unseen attributes, as well as their associated values. Experiments with totally 17,850 web pages in 4 verticals and comparison with RoadRunner [4] demonstrate the effectiveness of our framework.

The rest of this paper is organized as follows. Section 2 introduces related works. Section 3, 4 and 5 respectively present the construction of the proposed framework, and introduce details of the two procedures. Section 6 reports the experimental results, and section 7 makes the conclusion.

2. RELATED WORK

Towards the target of automatically extracting structured data from web pages, much research effort has been attracted. The most common information extraction technique for web pages is known as wrappers or templates, such as Adelberg’s work [1] and Liu et al.’s XWRAP [12]. Manually writing wrappers for web pages can precisely extract all the structured data, but it is the most tedious and time-consuming job in the past, as a wrapper must be built for each website, even each webstyle, and each time when webpage structure is updated.

Wrapper learning techniques can significantly reduce the amount of human efforts in constructing wrappers. Representative works include Turno’s adaptive framework [14] and Zheng et al.’s joint optimization method [18]. But as they are supervised methods, training examples are required to be collected and labeled. As a consequence, one restriction of a learned wrapper is that it cannot be applied to previously unseen sites.

To encounter this problem, generative models are extended to segment and label elements of web pages. For instance, Zhu et al. proposed two-dimensional CRFs [19] and hierarchical CRFs [20] respectively to predict semantics of web page elements based on individual elements and their inter-relationships in layout structures. Their limitation is that the set of possible attributes must be defined in advance with training samples for each attribute provided. Consequently, they can only extract attributes specified at training samples.

Several techniques such as interactive learning proposed by Irmak and Suel [9] and Kristjansson et al. [10] have been developed for reducing human efforts in preparing training samples. However, a substantial amount of human work is still required when learning wrappers from multiple websites.

There are also unsupervised template detection approaches based on a set of pages with similar layout formats to automatically extract their common attribute values (such as RoadRunner [4], and EXALG [2]) or identify repetitive patterns from a single page (with representative Liu and Zhai’s work [11, 17]). A major drawback of such methods lies in their disregard of semantics, and consequently great human effort must be paid on recognizing semantics of the extracted
data. Etzioni et al. revealed their KNOWITALL [6] and upgraded TEXTRUNNER [5], they are two unsupervised systems aiming at extracting relationships between entities from documents, but cannot applied to extracting attributes and attributes values associated with a particular record from a web page. An unsupervised framework with probabilistic graphical model is implemented by Wong et al. [16] to extract attributes and normalize their semantics from multiple web sites, but text fragments corresponding to attribute values of one page must be manually selected or highlighted to initialize its algorithm.

Recently, to lower human effort requirements, Wong et al. proposed a Bayesian approach to address a similar scenario [15]. Hao et al. [8] proposed a unified solution for structured web data extraction. Their framework is general to handle any vertical without re-implementation, and use only one labeled example sites for training to automatically deal with other sites in the same vertical. However, the effectiveness of their methods are subject to restrictions on the labeled website. Moreover, as there are large numbers of verticals, human efforts to label one website of each vertical need to be further reduced.

3. FRAMEWORK OVERVIEW
The basic flowchart of this framework is shown in Figure 2. There are mainly two procedures with five steps worked together to compose a dynamic learning framework for automatically extracting structured data from web pages without human efforts. The targets of these procedures and steps are summarized as follows.

![Figure 2: The flowchart of the proposed framework.](image)

(A) Credible Attributes Learning Procedure
This procedure aims at automatically attained, strictly filtered and dynamically updated the learning knowledge. Correspondingly, it is designed to gather potential items, to get candidate attributes, and then to generate credible attributes. This procedure is updated each time a page is added, thus gets more accurate results.

The items of structured data, as shown in Figure 1, such as ISBN-10, ISBN-13, publisher etc. are easily to be recognized by people. According to our observation, there are three kinds of common features shared by these data which contained in a large number of web pages:

**Structural Features:** most items are displayed in some well-structured formats in a web page which generally deliver both attribute and values to people. These formats can basically generalize to a kind of format like “attribute values”, from which we can find that the attribute always appear in the left side of values. Besides, these items are often concentrated in one or several fields of a page, which generally corresponds to an HTML element of this page.

**Inner-site Features:** for an attribute belongs to a vertical, if a site of this vertical contains this attribute, then it will appear in a considerable number of pages of this site, so we consider that the text fragments with low frequency in a site should not be an attribute.

**Cross-sites Features:** so many pages of a vertical share a limited number of common attributes, though they are very different in structure and other information. Most importantly, the number of sites in a vertical has no upper limit, while the number of attributes of a vertical is always no more than dozens. So there must be some attributes appear in pages of multiple sites.

1) **Gathering Potential Items** utilizes structural features of a web page to collect potential attributes and all their possible values.

2) **Getting Candidate Attributes** utilizes inner-site features to choose attributes that are commonly emerged in a website.

3) **Generating Credible Attributes** utilizes cross-site features to filter out incredible attributes, only keeping the most credible ones for further analysis. Thus, credibility of an attribute is defined and an algorithm based on the credibilities of attributes is employed to generate credible attribute set.

(B) Item Discovery and Extraction Procedure
Based on the credible attributes attained from the previous procedure, this procedure is implemented to detect item discovery fields, and to discover and extract existing credible and new unseen attributes, as well as their associated values.

1) **Detecting Items Discovery Fields** concerns about identifying the fields for attributes discovery and extraction, aims at filtering out parts of a page that don’t contain attributes. In this step, credible attributes are utilized as the clue and an algorithm is proposed to identify the field for attributes discovery.

2) **Discovering and Extracting Items** describes an algorithm in details that takes a field detected by the previous step as input and output items include attributes and values. Potential item set and their credibility values are used in the algorithm. Finally, the results of all discovery fields are integrated together as the final structured data extraction results we expected.
4. CREDIBLE ATTRIBUTES LEARNING

In this section, we are committed to describe the process of generating credible attributes from the stream of web pages. This process will be executed after a new page is added to the system and generate attribute information to support the process of discovering and extracting items from a page. As described in section 3, this task is implemented via three procedures working together, including extracting potential items, getting candidate attributes, and generating credible attributes. These procedures are described in the following three subsections, respectively.

4.1 Extracting Potential Items

According to the structural features of web pages, most attributes and corresponding values of entities, which are unified described as items in the paper, are generally displayed in well-structured formats for convenient information delivery for people. Based on this, three item structure models are provided for extracting nodes in DOM tree of a page who has well-structured attribute and values. It’s assumed that the attribute is in closely front of the values as shown in a page, and the attribute and values have a same ancestor node in the DOM tree.

The item structure models are shown in Figure 3, in which a rectangle illustrates only one text node and a circle is the parent of rectangles. Model (a) can be used to identify the case that an item consists of an attribute and a value, and the attribute node shares a same parent node with the value node in the DOM tree. Model (b) is used to find the parent of rectangles. Model (c) can explore multiple items at a time. In this model, attribute nodes and value nodes are presented alternately and all of them have a common parent node.

![Figure 3: Three item structure models.](image-url)

Of course, the item structure models shown in Figure 3 cannot depict all items of a page. In fact, we do not aspire to achieve this in this procedure. However, since there are enormous pages in the internet where we can learn enough knowledge about items. More specifically, there are plenty of pages from various websites in which the items of structured data can match these models, and thus makes the learning process goes well.

**Algorithm 1** Extracting Potential Items

**INPUT:** a DOM tree \( T \) of a new page, the item structure models set \( M \), and potential items information set \( \Pi \).

**OUTPUT:** updated potential items information set \( \Pi \).

```
for node \( t \) \in T do
    if \( t \) match \( m \in M \) then
        Convert \( t \) to items \( mi \);
        for item \( it \) \in \( m \) do
            if \( it.attribute \notin \Pi.attributeSet \) then
                \( \Pi \).addItem(\( it \));
            else
                \( \Pi \).getItem(\( it.attribute \)).values.add(\( it.values \));
            end if
        end for
    end if
end for
```

The potential item extraction algorithm, which is shown in Algorithm 1, aims at extracting potential items from a DOM tree and update the data related to these items (including attributes, values and site information) into the potential items information set \( \Pi \), which is empty when the learning task initially starts. \( \Pi \) is indexed by attribute, and the attribute values and sites information corresponding a certain attribute can be achieved by the index. With the given item structure model set \( M \) shown Figure 3 and the parsed DOM tree \( T \) of a new page, the algorithm checks whether a node \( t \) \in \( T \) matches an item structure model \( m \in M \). If matches, according to the meaning of model \( m \), attributes and values of items are extracted from \( t \), with another saying \( t \) is converted to items \( m \). Specifically, if \( t \) matches with model (a) or (b), it can be converted to one item; or if \( t \) matches with model (c), it can be converted to multiple items. The item information with \( m \) is appended to \( \Pi \) at last, as for an item \( it \) of \( m \), it will be added into \( \Pi \) if its attribute is not contained in the attribute set of \( \Pi \), otherwise the values of it are added to \( \Pi \).

4.2 Getting Candidate Attributes

Based on the inner-site feature mentioned in section 3, potential attributes with low frequencies in a site should be false positives. Besides, with the growth of the potential item information set \( \Pi \), it will be costly to learn credible attributes from such a huge number of potential attributes every time a new page added. Therefore, in our method, potential attributes with high frequencies in a site are gathered into a candidate attribute set \( A_{\text{cand}} \), which is used to learn credible attributes.

The frequency of an attribute \( a \) in site \( S_i \), \( i = 1, 2, \ldots \), is computed as:

\[
P_i(a) = \frac{N_i(a)}{|S_i|}
\]

(1)

where \( N_i(a) \) is the number of pages that contain the attribute \( a \) in site \( S_i \), and \( |S_i| \) is the count of recorded pages in site \( S_i \).

The frequency of an attribute \( a \) in site \( S_i \), \( i = 1, 2, \ldots \), is computed as:

\[
P_i(a) = \frac{N_i(a)}{|S_i|}
\]

where \( N_i(a) \) is the number of pages that contain the attribute \( a \) in site \( S_i \), and \( |S_i| \) is the count of recorded pages in site \( S_i \).
Algorithm 2 depicts the details of getting candidate attribute set. The inputs of this algorithm are a set of learned sites \( S \), and the potential item information set \( P_I \) maintained in the previous step. Firstly, the algorithm collects all attributes of every site \( s_i \in S, i = 1, 2, \ldots, |S| \). After that, it calculates the frequency of each attribute \( a \) in every site \( s_i \), and then compare it with a predefined threshold \( \theta \). An attribute \( a \) will be pushed into the candidate attribute set \( A_{\text{cand}} \), if its frequency is greater than \( \theta \).

\begin{algorithm}
\caption{Getting Candidate Attributes}
\begin{algorithmic}
\STATE \textbf{INPUT:} set of learned sites \( S \), potential items information set \( P_I \).
\STATE \textbf{OUTPUT:} candidate attribute set \( A_{\text{cand}} \).
\FOR{\( s_i \in S, i = 1, 2, \ldots, |S| \)}
\FOR{\( a_j \in P_I \).getAttributeSet(\( s_i \)), j = 1, 2, \ldots, \\text{do} \)}
\IF{\( P_I(a_j, \text{attribute}) > \theta \)}
\STATE \( A_{\text{cand}} .\text{add}(a_j) \);
\ENDIF
\ENDFOR
\ENDFOR
\end{algorithmic}
\end{algorithm}

4.3 Generating Credible Attributes

Although many potential attributes with low frequencies in a specific site have been filtered out when Getting Candidate Attributes, there may still be invalid attributes left in the candidate attribute set. To filter these invalid attributes, the cross-sites feature is considered to pick out credible attributes of a vertical via selecting the most frequent attributes in multiple sites of a specific vertical.

However, it’s not appropriate to justify whether a candidate attribute is a credible attribute just depends on its frequency. For instance, there are two attributes \( a_1, a_2 \) of a vertical, with \( a_1 \)’s frequencies \((0.5, 0.5)\) and \( a_2 \)’s frequencies \((0.9, 0.2)\) in two sites respectively. We believe that \( a_1 \) is more credible than \( a_2 \), as \( a_1 \)’s appearance in the two sites supports its significance, although it has a smaller sum of frequencies. Therefore, distribution of an attribute in multiple sites should be taken into account when we select credible attributes from candidate ones. In this paper, we employ the entropy to do so. The entropy of an attribute is defined as:

\[
\text{entropy}(a) = - \sum_{i=1}^{|S|} \text{DisP}_i(a) \log \text{DisP}_i(a)
\]  
\[(2)\]

where \( \text{DisP}_i(a) \) means the distribute proportion of attribute \( a \) in site \( s_i \) over all learned sites \( S \), and is specifically defined as:

\[
\text{DisP}_i(a) = \frac{P_i(a)}{\sum_{j=1}^{|S|} P_j(a)}
\]  
\[(3)\]

Finally, we evaluate the credibility of an attribute as follows:

\[
\text{credibility}(a) = \text{entropy}(a) + \sum_{i=1}^{|S|} P_i(a)
\]  
\[(4)\]

in which we have considered the frequency and distribution of an attribute in a vertical followed the description above.

Algorithm 3 shows how to use the credibility to find credible attributes. Given the candidate attributes set \( A_{\text{cand}} \) and a predefined number \( x \), the algorithm is composed of three phases. Firstly, build a credibility set \( A_{\text{cand}}.\text{CredibilitySet} \) through calculating credibilities of all attributes in \( A_{\text{cand}} \); secondly, set the average \( v \) of the most top \( x \) values in \( A_{\text{cand}}.\text{CredibilitySet} \) to be the threshold; lastly, all attributes with a credibility greater than the threshold \( v \) are added into the credible attribute set \( A_{\text{cred}} \).

The number of real attributes of a vertical is generally no more than a few dozens. Furthermore, the number of credible attributes is absolutely less than real attributes. So, as described in Algorithm 3, we decided to choose the top \( x \) attributes with highest credibility for generating credible attributes.

5. ITEMS DISCOVERY AND EXTRACTION

In this section, we introduce the details of extracting items from a page, which rely on the dataset generated from the credible attributes learning system, including the potential item information set and the credible attribute set. Since the credible attribute set may not contain all attributes of a new page, especially during the cold start of the system, a solution for discovering new attributes and their values is needed. As aforementioned in section 3, we divided the task of items discovery and extraction into two procedures, including detecting item discovery fields and discovering and extracting items.

5.1 Detecting Item Discovery Fields

The target for detecting item discovery fields is to select the text fragment fields from a page in which most items of the page are included. In fact, we can find that most items are displayed clustered in several fields of a page. Generally, the text fragments in a field correspond to the text nodes in a subtree of the DOM tree. So if we can confirm some attributes in a page, the fields contain these attributes also can be found by exploiting the features described above. Moreover, we can easily identify a text in a page as an attribute by matching it with the previously learned credible attribute set. Therefore, based on the credible attributes, an algorithm for detecting item discovery fields is shown in Algorithm 4.

Based on the feature previously described, we assume the text fragment which can be found in credible attributes set is an attribute and its next neighbor is the value of the item. Therefore, the CommonAncestor function in Algorithm 4,
which indicates the nearest common ancestor node of the two fragments, can present an item.

At the beginning, all text nodes in the DOM tree of a new page are listed in text nodes list \( L \) in order of their appearance in html source file. For a text node \( n_i \), there are two common cases when comparing \( n_i \) with all credible attributes. Case 1: \( n_i \).text equals one credible attribute. Then the parent node of common parent of \( n_i \) and its next sibling in the DOM tree are identified, and its text fragment list is considered as a discovery field. Case 2: prefix of \( n_i \).text equals one attribute. In this case, the text fragment list of \( n_i \).text is divided into two pieces by its prefix, and \( n_i \).text is replaced in the text fragment list of \( n_i \)’s parent node by these two text fragments. Rest cases will be ignored.

At the end, each discovery field in the set \( E \) would contain several credible attributes. However, it is not guaranteed that all text nodes in the discovery fields are valid attribute nodes, because other nodes may share the same parent node with attribute nodes.

### 5.2 Discovering and Extracting Items

The target of item discovery and extraction procedure is to discover attributes and values of items in a discovery field. In this paper, the discovery field consists of a list of text fragments and comes from the output of the previous procedure—Detecting Item Discovery Fields. And after this step, the results of item discovery and extraction in each field will be integrated as the final results for item extraction from the processed new page.

Recall that most items in a page are displayed together in a field, while some other information which is not attribute or value of items is also likely to be displayed. Based on the credibility of attributes, we develop an algorithm to complete the task of discovering and extracting items in a field, which summarized as Algorithm 5.

It is supposed that an attribute identifies an item. Therefore, the credibility of a text fragment in a field is calculated the same as credibility of an attribute, illustrated in Equation 4. As shown in Algorithm 5, if a text fragment has a higher credibility than the next text fragment and the threshold \( \theta \), it will be classified as the attribute of an item. Correspondingly, the next text fragment will be one of the values of the item.

According to the inner-site feature of attribute described in section 4.2, if an text fragment \( text \) can be identified as an attribute, the text must meet:

\[
\exists i, P_i(text) > \theta, \quad i = 1, 2, ..., |S|
\]

(5)

Thus it is guaranteed,

\[
\sum_{i=1}^{|S|} P_i(text) > \theta
\]

(6)

then according to the definition of credibility of attribute described in Equation 4, and \( \text{entropy}(text) \geq 0 \), we can get

\[
\text{credibility}(text) > \theta
\]

(7)

Therefore, the same threshold \( \theta \) with Algorithm 2 is set here.

However, an item is likely to have more than one values, so there should be a method for determining whether the next fragment of current value is another value of the item or not. In this paper, previous stored data of attribute values in the potential item information set are referred to solve this problem. Actually, values of some items may come from a limited lexicon (e.g., the Image resolution of camera) or contain some terms with high frequency (e.g., the manufacturer of auto). We filter out the values with frequencies lower than a minimum threshold \( \delta \), 0.002, for example. The text fragments not contained in \( \text{Enum}(\text{PI}.\text{getAllValues}(ca)) \) or with lower credibility than \( \delta \) will be dropped. Finally, to get full extracted attributes and corresponding values in a page, we integrate results of all discovery fields and ignore repeated ones.

### 6. EXPERIMENTAL EVALUATION
In this section, we report the evaluation results of our solution on real-world attributes extraction. Extensive experiments were performed on various websites of multiple verticals to evaluate our solution.

6.1 Experimental Setting
The dataset we used are derived from the dataset in experiments of [8]. 4 representative diverse verticals and around 17,850 web pages are used in this paper, including camera with 2,270 pages, book with 5,540 pages, job with 5,010 pages, university with 5,030 pages. And the pages of each vertical are uniformly distributed over 10 sites.

For evaluation, although [8] provided manually marked ground-truth results for the pages, but their extraction only focused on a small number of predefined attributes, thus didn’t provide whole information of an entity. For example, for books, they only extracted “title, author, publisher, ISBN13, and publish date”; other attributes like “language, format, dimensions, pages, age-range”, as shown in Figure 1, are ignored. Therefore, we remarked a new ground-truth that basically cover all the items of a page. Two human-accessors are invited to determine attributes and values of a page. If there is no describing attribute for a value, an appropriate attribute name will be assigned manually to this value. The extraction benchmark is the intersection from the two accessors, which means if there is a disagreement on a text selection, it will not be picked out. The selected pairs of attributes and values in each page are provided for evaluating the effectiveness of the method.

N-correct, precision, recall and $F_1$-measure are adopted for evaluation of our method. For each page, the N-correct of a method is the number of items, which are contained in standard data of this page. Precision of a method is the N-correct of this page, divided by total number of items, which are extracted by the method. The recall of a method is the N-correct of this page, divided by total number of items in standard data of this page. The $F_1$-measure is defined as the harmonic mean of equal weighting of pairwise recall and precision. As a side note, it is possible that a page contains repeated items. For this case, these items are treated as one.

6.2 Evaluation Effectiveness of Our Solution
Based on the dataset, we conduct an experiment to compare the effectiveness of our solution, which is to make use of an existing web pages data extraction approach called RoadRunner [4]. Notice that the data extracted by using RoadRunner are unlabeled, which means that only values are extracted, without their attributes. To compare the value given by RoadRunner to our ground-truth data, specifically, if the resulted value is the same with a value of ground-truth, we assign the ground-truth attribute to resulted value. Table 1 shows the comparison of our method to RoadRunner. The numbers not in parentheses indicate results of our approach, while the numbers in parentheses indicate results of RoadRunner. Each span column of the table corresponds to a run of the experiment on a vertical and each row refers to a site. The last row is the average performance.

From Table 1, we can find that the performance of RoadRunner is much lower than our framework for all the four verticals. The RoadRunner is based on the template similarity of multiple pages, thus can hardly consider cross-site features of a vertical. Though a lot of data can be obtained by RoadRunner, most of these data are meaningless false positives. One possible reason for this is that the web pages used here are real pages without preprocessing, thus containing noises such as advertisements and navigation banners, which greatly hamper RoadRunner’s performance. In contrast, our approach is more robust.

For our framework, it achieves a good performance for the camera and job verticals. And the results of university vertical would be satisfactory, if the result of site 8 (net-temps) is not considered. The reason that we extract nothing from site 8 is that the attributes in site 8 are different from other sites. As for the book vertical, some performance of our solution is not high, which is caused by the recommendation information in pages. In fact, consider that we didn’t previously clear the noise information of the pages (such as advertisement, navigation bar etc.), the performance even exceeds our expectation for job and university verticals. Since the attributes of the two verticals are relatively less, even a few errors can result in obvious decline in performance.

However, one of limitations of our framework is its requirement of attribute values appearing following their attributes. For example, the attribute value like the title of a book, in front of which there is no attribute text, can’t be extracted by our method. Some future work is considered to face this challenge. For instance, sets of attribute values can be analyzed to summarize regular expressions for further value recognition. Besides, to implement real automatic structured data extraction, we are also exploring automatic vertical classification method for webpages, as a preprocessing step for this framework.

7. CONCLUSIONS
In this paper, a dynamic learning framework to extract structured data from web pages without human effort is proposed. Firstly, by using structural features, inner-site features, and cross-sites features of web pages, credible attributes are generated in credible attribute learning procedure. Notice that this is a dynamic learning system, and thus the learned knowledge will be more accurate and reliable with the increasing of input web pages. Secondly, based on these credible attributes, the items discovery and extraction procedure is proposed to detect item fields, and to extract existing credible and unseen attributes, as well as their associated values. Different from other structured data extraction approaches which are static, require manually labeling samples and can not be flexible to unseen attributes, the characters of our approach is dynamic, automatic and its ability to fully extract structured data from web pages. Experiments based on a set with totally 17,850 web pages in 4 verticals demonstrated the effectiveness of our framework.

8. ACKNOWLEDGEMENT
This work is funded by the Natural Science Foundation of China (NSFC, Grant Nos. 60873237 and 61003168), Natural Science Foundation of Beijing (Grant No.4092837), Outstanding Young Teacher Foundation and Basic Research Foundation of Beijing Institute of Technology, and partially supported by Beijing Key Discipline Program.
Table 1: The extraction performance on the verticals of camera (CAM), book (BOOK), job (JOB) and university (UNI). N:N-correct, P:precision, R:recall, F:F1-measure

<table>
<thead>
<tr>
<th></th>
<th>CAM</th>
<th>BOOK</th>
<th>JOB</th>
<th>UNI</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>N</td>
<td>P</td>
<td>R</td>
<td>F</td>
</tr>
<tr>
<td>1</td>
<td>6.7</td>
<td>0.95</td>
<td>0.76</td>
<td>0.76</td>
</tr>
<tr>
<td>2</td>
<td>58.6</td>
<td>0.91</td>
<td>0.84</td>
<td>0.87</td>
</tr>
<tr>
<td>3</td>
<td>32.5</td>
<td>0.82</td>
<td>0.53</td>
<td>0.63</td>
</tr>
<tr>
<td>4</td>
<td>36.5</td>
<td>0.94</td>
<td>0.77</td>
<td>0.85</td>
</tr>
<tr>
<td>5</td>
<td>6.9</td>
<td>0.94</td>
<td>0.67</td>
<td>0.77</td>
</tr>
<tr>
<td>6</td>
<td>3.9</td>
<td>0.66</td>
<td>0.39</td>
<td>0.48</td>
</tr>
<tr>
<td>7</td>
<td>27.2</td>
<td>0.91</td>
<td>0.74</td>
<td>0.81</td>
</tr>
<tr>
<td>8</td>
<td>67.5</td>
<td>0.96</td>
<td>0.88</td>
<td>0.92</td>
</tr>
<tr>
<td>9</td>
<td>50.7</td>
<td>0.88</td>
<td>0.84</td>
<td>0.86</td>
</tr>
<tr>
<td>10</td>
<td>39.6</td>
<td>0.89</td>
<td>0.92</td>
<td>0.90</td>
</tr>
<tr>
<td>Avg.</td>
<td>35.2</td>
<td>0.88</td>
<td>0.72</td>
<td>0.72</td>
</tr>
</tbody>
</table>

9. REFERENCES


