From One Tree to a Forest: a Unified Solution for Structured Web Data Extraction

Qiang Hao[†‡][*], Rui Cai[†], Yanwei Pang[‡], Lei Zhang[†]
[†]Microsoft Research Asia, Beijing 100080, P.R. China
[‡]Tianjin University, Tianjin 300072, P.R. China
[‡]{qhao, pyw}@tju.edu.cn, [†]{ruicai, leizhang}@microsoft.com

ABSTRACT

Structured data, in the form of entities and associated attributes, has been a rich web resource for search engines and knowledge databases. To efficiently extract structured data from enormous websites in various verticals (e.g., books, restaurants), much research effort has been attracted, but most existing approaches either require considerable human effort or rely on strong features that lack of flexibility. We consider an ambitious scenario – can we build a system that (1) *is general enough to handle any vertical without re-implementation* and (2) *requires only one labeled example site from each vertical for training to automatically deal with other sites in the same vertical*? In this paper, we propose a unified solution to demonstrate the feasibility of this scenario. Specifically, we design a set of weak but general features to characterize vertical knowledge (including attribute-specific semantics and inter-attribute layout relationships). Such features can be adopted in various verticals without redesign; meanwhile, they are weak enough to avoid overfitting of the learnt knowledge to seed sites. Given a new unseen site, the learnt knowledge is first applied to identify page-level candidate attribute values, while inevitably involve false positives. To remove noise, site-level information of the new site is then exploited to boost up the true values. The site-level information is derived in an unsupervised manner, without harm to the applicability of the solution. Promising experimental performance on 80 websites in 8 distinct verticals demonstrated the feasibility and flexibility of the proposed solution.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval - Information filtering; I.5.1 [**Pattern Recognition**]: Models - Statistical

[*]This work was performed at Microsoft Research, Asia.

General Terms

Algorithms, Experimentation, Performance

Keywords

Structured data, information extraction, vertical knowledge, site-level information

1. INTRODUCTION

With the development of search engines and knowledge databases [2], the target for indexing and retrieval goes beyond keywords to more concrete and comprehensive information, which is usually represented by structured data. Structured data is in the form of entities and associated attributes, *e.g.*, a `book` and its attributes {`title`, `author`, `publisher`}. Various categories of entities, called verticals, have been involved in a variety of applications. For example, for product search, it needs to collect product information for verticals like `Book`, `Auto`, and `Camera`. In addition, in each vertical, there are usually tens of thousands websites containing related structured data. To collect structured data from the Web, an industrial solution is desired to be *flexible* to handle various vertical, *scalable* to deal with numerous websites in each vertical, and *automatic* enough to lower down human effort.

Towards such a practical solution, we consider an ambitious scenario in this paper. That is, to build a system which (1) *is general enough to handle any vertical without re-implementation* and (2) *requires only one labeled example site from each vertical for training to automatically deal with other sites in the same vertical.*

However, it is a non-trivial task to extract structured data from unseen sites based on limited knowledge from only one labeled seed site. Consider the example shown in Fig. 1, where three verticals with associated attributes are listed to the left, as the target of structured data extraction. For each vertical, a seed site is labeled to indicate the occurrences of each attribute in its pages, based on which we aim to identify attribute values from web pages on other sites. The challenges in the task fall into three aspects:

– **Variation of attribute values.** Different sites usually deliver different values of the same attribute, leading to the difficulty in identifying unseen values. Essentially two reasons account for such inter-site variation. Firstly, some attributes (*e.g.*, `book-title`) correspond to numerous and diverse values which appear in different sites unevenly and can hardly be fully covered by a seed site. Secondly, attribute values may be presented using site-specific formats

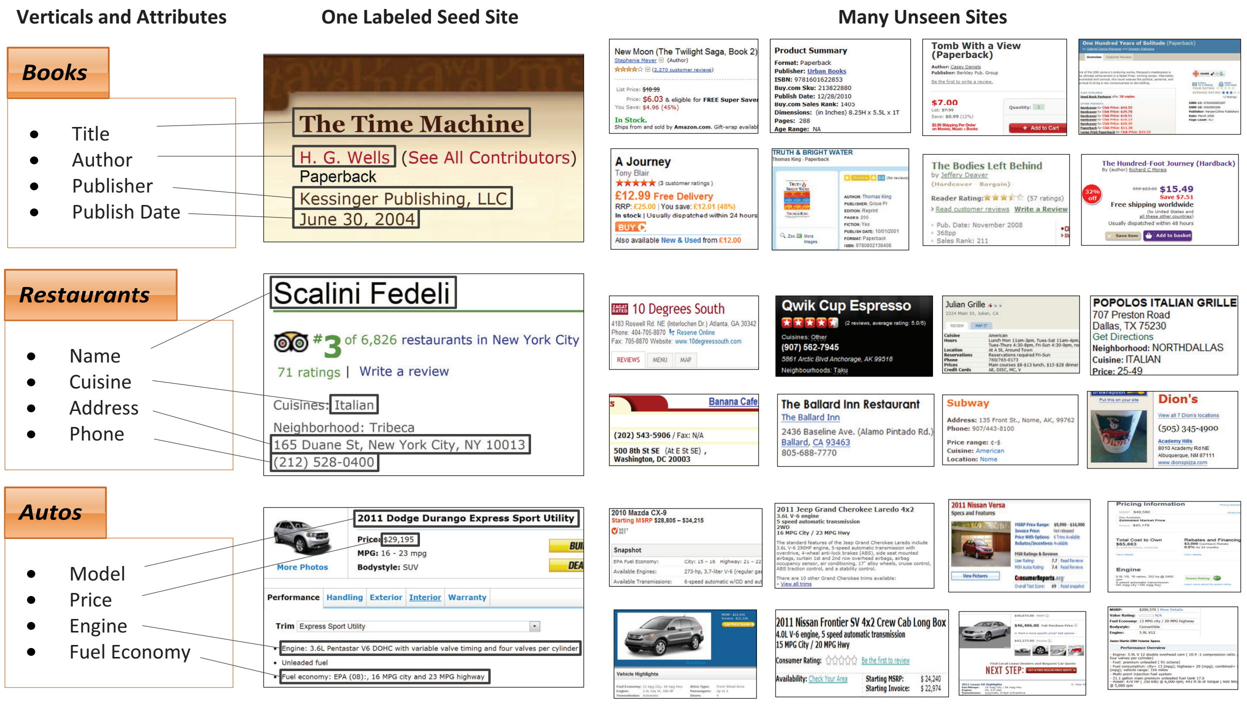

Verticals and Attributes
One Labeled Seed Site
Many Unseen Sites
Books
Title
Author
Publisher
Publish Date
The Time Machine
H. G. Wells (See All Contributors)
Paperback
Kessinger Publishing, LLC
June 30, 2004
Restaurants
Name
Cuisine
Address
Phone
Scalini Fedeli
#3 of 6,826 restaurants in New York City
71 ratings | Write a review
Cuisines: Italian
Neighborhood: Tribeca
165 Duane St, New York City, NY 10013
(212) 528-0400
Autos
Model
Price
Engine
Fuel Economy
2011 Dodge Durango Express Sport Utility
Price: $29,195
MPG: 16 - 23 mpg
Bodystyle: SUV
Engine: 3.6L Pentastar V6 DOHC with variable valve timing and four valves per cylinder
Fuel economy: EPA (08):, 16 MPG city and 23 MPG highway

Figure 1: An illustration of the proposed scenario: automatically extracting structured data from websites based on one labeled seed site per vertical.

(*e.g.*, `book-author` can be presented as either abbreviated or full names).

– **Variation of layout.** In each vertical, the attributes are usually presented by different sites in various manners (*e.g.*, not necessarily in consistent positions or in well-formatted structures like tables), due to the various page layout structures across sites.

– **Noisy page content.** Web pages usually contain much noisy content intertwined with the target attribute values to be extracted. For instance, a book page may contain multiple date-type content besides the value of attribute `publish-date` we target on.

Related to the scenario we are concerned with, much research effort has been dedicated to extracting structured data from web pages. One popular category of methods relies on the DOM-tree representation of web pages [1], as well as the fact that many web pages are generated based on templates, which wrap data records in HTML tags. One early stream of such template-based work is wrapper induction [9,11,12,18], which relies on manual annotation on example pages of a template and learns wrappers used for data extraction from other pages of that template. These methods can hardly scale up to large number of sites due to the considerable human effort required for both labeling unseen sites and maintaining wrappers of observed sites with periodically updated templates. To alleviate human labor, some unsupervised methods are proposed to extract structured data automatically, by deducing a template from a set of pages [4,7] or by identifying repetitive patterns from a single page [6,10,17]. One major drawback of such methods lies in the disregard of semantics and consequent requirement for human effort to identify attribute values from all the extracted content.

Another category of methods further consider semantics during automatic data extraction by exploiting the content and page layout features (*e.g.*, positions, font sizes), other than HTML tags, of web pages in a unified manner. For instance, extensions of CRFs (conditional random fields) are proposed to predict semantics of web page elements based on individual elements and their interrelationships in two-dimensional [19] or hierarchical [20] layout structures. However, such CRF-based methods are not general enough to handle various verticals, but rather rely on carefully designed features specific to a target vertical. The semantic features adopted in [5] are more general to handle various verticals, but it still needs to train the model based on a number of example pages from various websites to avoid overfitting, leading to labor-intensive burden of preparing training data. Detailed comparisons of our method and [5] will be introduced in the experiments.

In this work, we aim to figure out a unified solution for structured web data extraction, with flexibility to various verticals and minimal human effort for labeling examples. To achieve this goal, we combine the information considered by above two categories of existing approaches, that is, to (a) exploit content and layout structures of pages and simultaneously (b) take advantage of interrelationships between template-generated pages. On one hand, we propose to derive relatively weak but general features from page content and layout structures. Such features keep capability of characterizing vertical knowledge learnt from a seed site to identify attribute values from new sites. In contrast to vertical-specific strong features, these features can be applied in various verticals without redesign; meanwhile, they are weak enough to avoid overfitting of the learnt knowledge to seed sites. On the other hand, to compensate the possible performance loss compared with solutions relying on strong features, we boost the preliminary page-level results by exploiting site-level information, based on the assumption that there are strong interrelationships between pages from the same website.

In general, site-level information leads to two main benefits as follows.

- **Providing useful features.** Based on site-level statistics, we can derive features useful to identify true attribute values and filter out noise. For instance, duplicate elements among pages can be identified by analyzing inter-page redundancy, and serve as contexts of their adjacent elements to provide semantic hints.
- **Boosting page-level attribute identification.** Site-level information can serve as inter-page constraints to boost page-level identification of attribute values. Because an incorrect decision on a page element could get revised by referring to the decisions made on many corresponding elements on other pages (*e.g.*, via a majority voting). In this way, occasionally identified false positives on some pages can get removed, while the true positives which receive sufficient supports from many pages are boosted up.

Moreover, such site-level information is derived in an unsupervised manner from each website, without involving burden of additional human labor.

Based on the above ideas, a unified solution is proposed in this paper for automatically extracting structured data for an arbitrary vertical and associated attributes. Specifically, we design a set of general features to characterize vertical knowledge, including attribute-specific semantics and inter-attribute layout relationships, derived from a seed site with labeled attribute values. For a given new site in the same vertical, the learnt vertical knowledge is first applied to identify candidate attribute values at the page level. As the knowledge is relatively weak, such page-level results inevitably contain false positives. To further remove noise, site-level information of the new site is then exploited to boost up the true values in an unsupervised way. We evaluated the proposed solution on 80 websites in 8 distinct verticals and obtained promising results, verifying the feasibility and flexibility of the proposed solution.

The rest of this paper is organized as follows. We first refer to related work in Section 2, and outline the proposed solution in Section 3. The algorithm details are introduced from Section 4 to Section 6. Experimental results are reported in Section 7. In Section 8, we conclude the paper and discuss some future directions.

2. RELATED WORK

Extracting structured data from web pages has been studied extensively. An early stream of research focuses on wrapper induction [9, 11, 12], which relies on manually annotated examples to learn data extraction rules. A more recent work [18] further combines template detection with wrapper generation to improve extraction accuracy. Such approaches are semi-automatic and difficult to be scaled up, due to the human labor required for labeling pages of new sites. Another category of automatic approaches deduce templates from web pages sharing identical format. For instance, Roadrunner [7] generates a template by comparing the current template with more pages, while EXALG [4] deduces templates by analyzing co-occurrence of tokens across pages. One common limitation of these approaches is that they do not concern the semantics of extracted data; whereas we aim to extract structured data corresponding to specified semantics (i.e., vertical attributes).

To better leverage sematic information, a series of research efforts have been dedicated to content analysis of web pages. In [8], keyword distributions are learnt from an existing knowledge base and used to annotate web pages. Besides prior knowledge, machine learning techniques are also utilized. For instance, Zhu et al. [19, 20] propose CRF-based probability models to exploit the contents and page layout features of web pages for data record detection and semantic labeling. These methods rely on either existing vertical specific knowledge or abundant training data, and thus are not suitable for our scenario where flexibility for various verticals and minimal human effort are desired.

In recently years, some approaches are proposed to extract information within specific verticals. In [16], structured data in web forums is extracted by exploiting site-level knowledge, including interrelationships between homogeneous pages and linkages between heterogeneous pages. In [15], the authors present an unsupervised approach to extract information from online job advertisement documents in different domains. In contrast to these works, our work targets on a general solution for various verticals.

We also noticed that there is a very recent work [14] which addresses a similar scenario of learning wrappers from one website and adapting them to other sites. In [14], the authors focus on generative probabilistic model-based machine learning. By contrast, we are concerned more with the exploitation of site-level information and DOM structures than sophisticated learning models. Although our lightweight solution only adopts off-the-shelf learning techniques, the performance is still competitive with that reported in [14][1].

3. FRAMEWORK OVERVIEW

In our scenario, a specific task takes as input a vertical (*e.g.*, `book`) associated with a set of attributes and a labeled seed site, and aims to extract attribute values from an unseen site in the same vertical. By representing each web page as a DOM tree, the task is essentially to identify the *text nodes*, *i.e.*, leaf nodes with text-type values in DOM trees, that contain values of each target attribute.

The basic flowchart of the proposed solution is shown in Fig. 2. Given a labeled website as seed, we first extract features from its pages and learn vertical knowledge. Then, given a site previously unseen, we randomly sample a set of pages for the sake of efficiency in subsequent processing. After extracting features from these sample pages, the learnt knowledge is applied to the new site to identify attribute values. Such extraction results further serve as substitutes of manually labeled examples required in wrapper induction [18], to generate wrappers for structured data extraction from the entire page collection of the new site. Optionally, together with the extracted attribute values, the new site could serve as another "labeled" seed to provide additional vertical knowledge, in a bootstrapping manner.

[1]The extensive comparisons to the above state-of-the-art methods (*e.g.*, [14]) are in our future plan. It still needs considerable efforts to implement these algorithms.

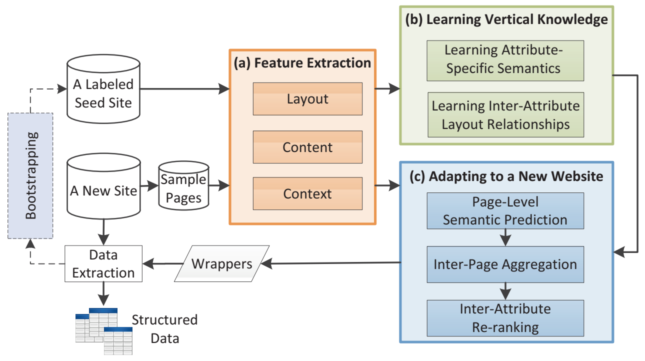

Bootstrapping
A Labeled Seed Site
A New Site
Sample Pages
Data Extraction
Structured Data
Wrappers
(a) Feature Extraction
Layout
Content
Context
(b) Learning Vertical Knowledge
Learning Attribute-Specific Semantics
Learning Inter-Attribute Layout Relationships
(c) Adapting to a New Website
Page-Level Semantic Prediction
Inter-Page Aggregation
Inter-Attribute Re-ranking

Figure 2: The flowchart of the proposed solution, which consists of three components: (a) feature extraction, (b) learning vertical knowledge, and (c) adapting to a new website.

As shown in Fig. 2, the algorithms of the solution are mainly distributed in three components, namely *feature extraction*, *learning vertical knowledge*, and *adapting to a new website*. These components are summarized as follows.

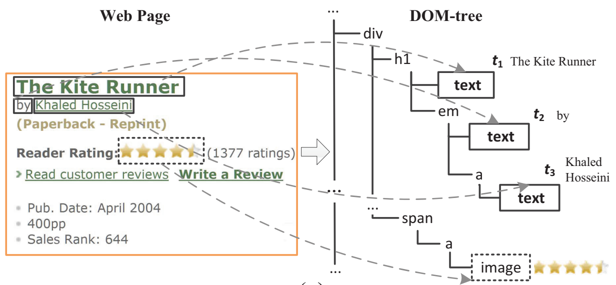
- **Feature extraction** takes multiple pages from a site as input and output a set of text nodes associated with ***content***, ***context***, and ***layout*** features. Details of this component are introduced in Section 4.
- **Learning vertical knowledge** learns two kinds of vertical knowledge from a seed site based on its attribute labels and extracted features. Knowledge of ***attribute-specific semantics*** is derived from *content* and *context* features for each attribute, while knowledge of ***inter-attribute layout*** is derived from *layout* features. Details of this part are presented in Section 5.
- **Adapting to a new website** exploits the learnt vertical knowledge to identify attribute values from a new site in three steps. In ***page-level semantic prediction***, the learnt *semantic* knowledge is applied to predict the semantics of text nodes in each page. In ***inter-page aggregation***, the page-level results are aggregated at the site level to get revised. In ***inter-attribute re-ranking***, the learnt *layout* knowledge is utilized to figure out the optimal results from both the semantic and layout perspectives. This component is described in Section 6.

4. FEATURE EXTRACTION

As described in Section 3, in this paper, we focus on identifying attribute values contained in text nodes. Other types of leaf nodes in DOM trees (*e.g.*, the image node representing rating score in Fig. 3 (a)) are out of our scope. For a set of pages from a given site, we parse them into DOM trees to obtain text nodes, from which three categories of features are extracted, including *layout features*, *content features* and *context features*. These features are described in the following three subsections, respectively.

4.1 Layout Features

Layout features describe the properties of a text node as a visual element on screen, when it is rendered via a web browser (*e.g.*, Internet Explorer). Layout information can help characterize both the intra-page and inter-page relationships between text nodes.



(a)

Node	Dom Path	Visual Size	Visual Position
t_1	/html/body/div/div/div/div/h1/text	(256, 32)	(0, 766)
t_2	/html/body/div/div/div/div/h1/em/text	(24, 16)	(0, 798)
t_3	/html/body/div/div/div/div/h1/em/a/text	(120, 16)	(24, 798)

(b)

Figure 3: Illustrations of (a) text nodes in a DOM tree and (b) layout features associated with a text node: DOM path, visual size and visual position.

In this paper, we leverage three types of layout features listed as follows.

- **DOM path** of a text node t, denoted by $dom(t)$, is its corresponding root-to-leaf tag path, which roughly characterizes its location in the DOM tree.
- **Visual size** of a text node t is a pair $\langle width(t), height(t)\rangle$ namely the width and height (in pixels) of its corresponding bounding box[2] on screen in web page rendering.
- **Visual position** of a text node t is the 2-D coordinates of the top-left vertex of its bounding box, denoted as another pair in pixels $\langle left(t), top(t)\rangle$.

For illustration, Fig. 3 (b) lists some example layout features of the text nodes in Fig. 3 (a).

4.2 Content Features

Content features correspond to the textual values contained in text nodes, and are very useful for characterizing the semantics of text nodes. In this paper, the value contained in a text node t is denoted by v_t, based on which five types of content features are extracted.

Unigram set. There are a number of attributes corresponding to values that come from a limited lexicon (*e.g.*, `NBA team`) or contain some representative terms with high frequencies (*e.g.*, the term "press" is very common in values of `book publisher`). Intuitively, pre-defined attribute-specific lexicons can help identify such kind of attributes, but it is labor-intensive to build lexicons manually for numerous attributes within various verticals. Instead, we leverage unigram as a general feature to characterize such information. Specifically, the *unigram set* of a text node t is defined as the set of tokens in v_t,

$$unigram(t) = \{w | w \in v_t\},$$

where v_t is split on whitespace into tokens $\{w\}$.

Token count. The values of many attributes consist of a relatively fixed number of tokens. For instance, `person`

[2]The bounding box of a DOM node can be retrieved using programming interfaces provided by web browsers, such as the APIs for Internet Explorer [3].

`name` usually contains 2 $\sim$ 4 terms. To enable learning of such knowledge from a labeled website, we define the *token count* of a text node t as a numerical feature,

$$token\_count(t) = |unigram(t)|.$$

Character count. It is observed that sometimes token-level granularity is still too coarse to distinguish between values of different attributes. For instance, both `ISBN-13` and `ISBN-10` of `book` contain only one token, while `ISBN-13` has 13 digits and `ISBN-10` has only 10. Hence, complementary to token count, a feature in a finer granularity is required to characterize the length of textual values. Here, we adopt the *character count* of a text node t,

$$char\_count(t) = \sum_{w \in unigram(t)} number\_of\_characters(w).$$

Character type. A variety of attributes correspond to values composed of specific types of characters. For example, values of `price` often contain digits and symbols (*e.g.*, \$, £). The occurrences of different character type are descriptive to such attributes, and are utilized by some existing work in a separate way (*e.g.*, directly using the occurrences of some specific characters as features). To be flexible to various attributes, we focus on three general character types (*i.e.*, *letter*, *digit*, and *symbol*) and define the *character type* of a text node t as a vector that encodes the proportion of each character type among all non-whitespace characters in v_t,

$$char\_type(t) = (c_t^{letter}, c_t^{digit}, c_t^{symbol}),$$

where c_t^* is the proportion of corresponding character type.

Page redundancy. We also observe that some attributes have distinct value redundancies. For instance, in the vertical `restaurant`, the attribute `cuisine` has very redundant values while `name` has almost unique values. Therefore, value redundancies of text nodes among pages are potentially helpful for predicting their semantics. Specifically, we define the *page redundancy* of a text node t as

$$page\_redundancy(t) = |\mathcal{P}_{v_t}| \,/\, |\mathcal{P}_{all}|,$$

where $\mathcal{P}_{v_t}$ is the set of pages that contain at least one text node with the value v_t, and $\mathcal{P}_{all}$ is the entire set of input pages. Note that the numerator is defined analogous to the document-frequency (DF) of a term in information retrieval, since what we concern is the inter-page redundancy rather than intra-page duplicates.

4.3 Context Features

In contrast to layout features and most content features, context features exploit site-level information to capture the surrounding text indicating semantics of text nodes. Actually, we have observed on many websites that the presented attribute values are preceded by some text that indicates their semantics (*e.g.*, "*Publisher:*" preceding to a `book publisher` value). This kind of text serves as site-level context to co-occur with corresponding attribute values extensively across pages, and thus can help identify values of such attributes if properly exploited.

Some existing approaches identify attribute values by seeking matches between handcrafted context lists and surrounding text of text nodes. However, these attribute-specific context lists require much human effort to create. Therefore, it

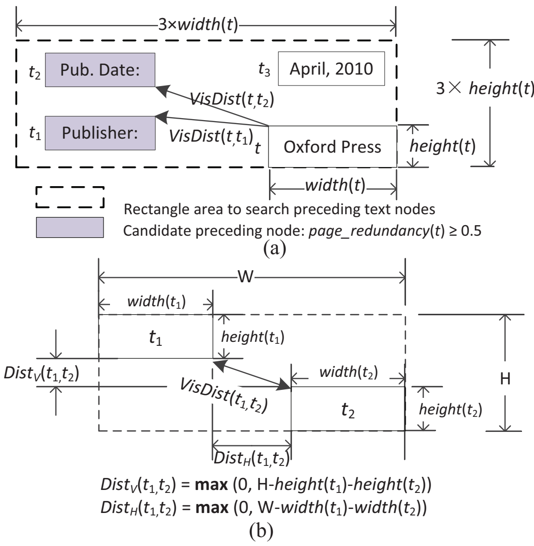


Figure 4: Illustrations of (a) identifying the preceding text of a text node t and (b) calculating the visual distance $VisDist(t_1, t_2)$ between two text nodes t_1 and t_2 when they are rendered on screen.

is highly desired to design features that automatically capture such context as semantic clues. Besides, we also notice that the values of some attributes tend to contain common prefixes and/or suffixes, which potentially indicate and characterize the semantics of attribute values. Although such prefixes/suffixes are actually parts of text node values, we take them as a type of context because they serve as site-level semantic clues. Based on the above considerations, we propose two types of context features as described below.

Preceding text of a text node t, denoted by $preceding(t)$, is defined as the value of its preceding text node t^* that is constrained to be:

– located to the top-left of t and within a rectangle proportional to the visual size of t, as shown in Fig. 4 (a).
– associated with high *page_redundancy* (exceeding an empirical threshold such as 0.5). Here the preceding node is required to be sufficiently redundant among pages, because the context is expected to be static at the site level.
– more close to t than any other candidates that meet the previous two criteria (higher *page_redundancy* is preferred for the case of equal distance). As illustrated in Fig. 4 (b), the visual distance between two nodes is defined as

$$VisDist(t_1, t_2) = \sqrt{Dist_H(t_1, t_2)^2 + Dist_V(t_1, t_2)^2}.$$

For the example case in Fig. 4 (a), t_1 is determined to be the preceding node of t, yielding the preceding text "*Publisher:*" for the value "Oxford Press". When no preceding text node identified, $preceding(t)$ is defined as null.

Prefix and Suffix of a text node t, denoted by $prefix(t)$ and $suffix(t)$, are defined as the beginning and ending substring (within its value v_t) commonly shared by lots of text nodes across pages. Given a set of text nodes coming from multiple pages, we extract their prefix and suffix features in three steps as follows.

1. Group values of text nodes in terms of rough formats, which are obtained by summarizing successive letters

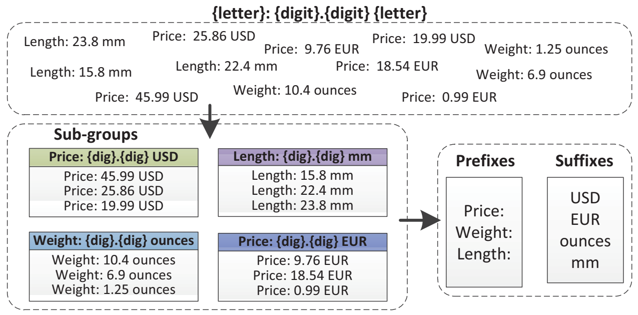


Figure 5: An illustration of identifying prefixes and suffixes. Text strings are grouped according to their formats, based on which popular sub-strings are discovered as prefixes or suffixes.

and digits with wildcards {letter} and {digit} (*e.g.*, the format of "2.36 cm" is "{digit}.{digit} {letter}").

2. For each group, further cluster text nodes according to their specific sub-strings.
3. Prefixes and suffixes are identified based on those sub-groups which contain text nodes from a sufficient proportion (*e.g.*, 50%) of pages.

As an example shown in Fig. 5, a set of values in the format "{letter}: {digit}.{digit} {letter}" are categorized into four sub-groups to extract prefixes and suffixes.

5. LEARNING VERTICAL KNOWLEDGE

In this section, we describe the detailed methods used to learn vertical knowledge from a seed site. Given a specified vertical and associated set of m attributes, denoted as $\mathcal{A} = \{a_1, \ldots, a_m\}$, we aim to learn two kinds of, namely *attribute-specific* and *inter-attribute*, vertical knowledge. For attribute-specific knowledge, we mainly focus on characterizing the semantics of each single attribute $a_j \in \mathcal{A}$ based on *content* and *context* features extracted from text nodes labeled with a_j. For inter-attribute knowledge, we target on characterizing the layout relationships between attribute based on the *layout* features of corresponding text nodes. The learning of such two kinds of knowledge is detailed in the following two subsections, respectively.

5.1 Learning Attribute-Specific Semantics

Given a set of text nodes from a labeled seed site, we learn the semantic knowledge of each attribute $a_j \in \mathcal{A}$ from a subset $\mathcal{T}_j$ of text nodes that are labeled as containing values of a_j. A straightforward idea is to combine all the features directly using a unified model. However, the *content* and *context* features have quite different behaviors – content features provide statistic evidences that can be adopted to evaluate any input text; while context features show indicative hints which are sufficient but not necessary. More specifically, matching of context features gives a strong confidence to acknowledging an attribute, but mismatching does not mean denying since context of an attribute may vary across websites. Therefore, we propose to exploit content and context features separately, to benefit from both of them.

For each attribute a_j, a classifier is trained to incorporate all the five types of content features described in Section 4.2. As different content features are heterogeneous and with different scales, each feature is first normalized into $[0, 1]$. For the *unigram set*, frequencies of tokens are first counted from $\{unigram(t) \mid t \in \mathcal{T}_j\}$, and then each unigram set is normalized based on the average frequency of its element token(s). For the *character type* with vector values, each feature value is converted to the cosine similarity to the mean of feature values in $\{char\_type(t) \mid t \in \mathcal{T}_j\}$. And for the other three content features with scalar values, the normalization is simply based on their maximums and minimums. After the normalization, an SVM classifier is built to learn a mapping from the content features of a text node t to an estimated probability, $ContentRel(t \,|\, a_j)$, that t is relevant to a_j.

For each attribute a_j, three lookup tables are respectively constructed for the *preceding text*, *prefix*, and *suffix* features described in Section 4.3. Each lookup table consists of feature values aggregated from text nodes in $\mathcal{T}_j$. For an unseen text node t, its context-based relevance to a_j, $ContextRel(t \mid a_j)$, is defined as the percentage of t's context features matching the corresponding lookup table of a_j. Here, exact match is adopted for the sake of efficiency, while more sophisticated measurements (*e.g.*, edit distance) are also feasible choices.

To combine the clues of both content and context features, the overall relevance of a text node t to an attribute a_j is estimated by

$$Rel_{TN}(t \,|\, a_j) = \max(ContentRel(t \,|\, a_j), ContextRel(t \,|\, a_j)).$$

The **max** operation is adopted here to properly leverage context features. As mentioned above, context features are sufficient but not necessary. Hence, we trust context features if the corresponding score is high enough; otherwise, the relevance estimation should rely only on content features.

5.2 Learning Inter-Attribute Layout

The power of page layout structures in identifying attribute values has been demonstrated by existing approaches such as [19]. In contrast to existing solutions that couple layout and semantic information together, we treat inter-attribute layout and attribute-specific semantics separately. This is because the attribute layout tends to vary across sites and should be exploited in a light-weighted manner.

In this paper, the inter-attribute layout is characterized just using pairwise visual distances. That is, given a set of m attributes and a seed site with attribute labels, an $m \times m$ layout matrix $\mathbf{A}$ is constructed, in which each element $\mathbf{A}_{ij}$ encodes the average visual distance between the attributes a_i and a_j, as

$$\mathbf{A}_{ij} = \frac{1}{w_{screen}} \cdot \frac{\sum VisDist(t_{a_i}, t_{a_j})}{|\mathcal{P}_{a_i \bigcap a_j}|},$$

where t_{a_i} and t_{a_j} are two text nodes labeled with a_i and a_j respectively on a page; $\mathcal{P}_{a_i \bigcap a_j}$ is the set of all pages containing values of both a_i and a_j; and $VisDist(\cdot, \cdot)$ is the same one defined in Section 4.3. To handle the inter-site variations of page size, the average distance is further normalized by the whole rendering width of a page, w_{screen}, which is almost consistent for every page in the same website.

An illustrative example is given in Fig. 6, where the inter-attribute layout matrices of five websites in the `book` vertical are visualized. From Fig. 6, it is clear that there are much consistence between layout matrices of these five websites (*e.g.*, `title` and `author` are closed to each other), despite some minor variations.

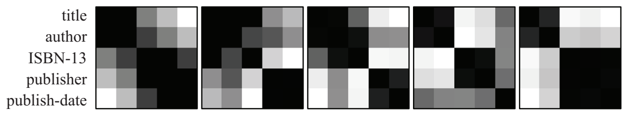


Figure 6: An illustration of the layout matrix A for five websites from the book vertical. The darker the matrix element, the closer the corresponding attributes in rendering.

6. ADAPTING TO A NEW WEBSITE

In this section, we introduce the details of adapting the learnt knowledge to extract structured data from an unseen website. As aforementioned, the learnt knowledge is weak and is just capable of indicating all possible candidate answers, a major part of which consists of noisy false positives. We will show in this section how the site-level statistics can help filter the candidates and boost the true answer. For the sake of efficiency, the adaptation is based on a set of pages $\mathcal{P}_{sample}$ randomly sampled from the new website.

6.1 Page-Level Semantic Prediction

Given a page $p \in \mathcal{P}_{sample}$, we extract its text nodes as well as the content and context features for each node. Following Section 5.1, each text node t is associated with an m-dimensional vector

$$(Rel_{TN}(t\,|\,a_1), \ldots, Rel_{TN}(t\,|\,a_m)),$$

which indicates its relevance scores to all the attributes. To determine the value of an attribute a_j on the page p, the simplest way is to assign it with the text node t_{a_j}

$$t_{a_j} = \arg\max_t Rel_{TN}(t\,|\,a_j).$$

6.2 Inter-Page Aggregation

The page-level results of semantic prediction are inevitably not accurate enough, due to the inter-site variations and weak features used to characterize vertical knowledge. Therefore, we resort to site-level information to boost these preliminary results, by exploiting inter-page dependencies on the new site. In other words, the semantics of some text nodes are highly consistent to some other text nodes in other pages, and thus we can get more accurate predictions by considering them simultaneously.

To leverage such inter-page dependencies, aligning text nodes in different pages is a crucial issue. A popular method is partial tree alignment [17], which performs partial matching between DOM trees. However, partial tree alignment is not appropriate for our task due to some reasons. First, the pairwise alignment of DOM trees is quite time-consuming, even for hundreds of sample pages. Furthermore, the alignment only considers information of HTML tags, and thus cannot differentiate between text nodes with distinct semantics but identical DOM paths[3]. Fig. 7 shows some statistics on 30 websites in three verticals. It is clear that DOM paths severely lack the ability to distinguish between text nodes.

Therefore, semantic information is required to distinguish between text nodes in different roles. Here, we propose to leverage the context features presented in Section 4.3. That is, two text nodes can be aligned if they have both the same

[3]This is common in a table-like structure, where table cells share almost the same HTML tags but very likely have different content.

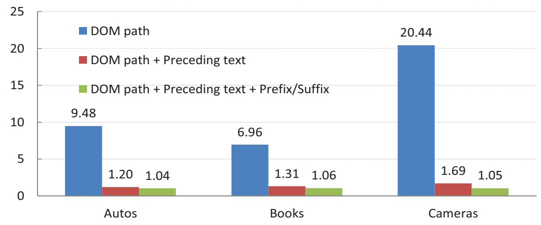


Figure 7: The average numbers (per page) of text nodes that have identical DOM path, DOM path+preceding text, and DOM path+preceding text+prefix/suffix, on websites in three example verticals.

DOM path and context features. From Fig. 7, it can be seen that context features could help lower down the risk of misalignment significantly. A set of aligned text nodes is called a *data field*, denoted as D. The detailed algorithm of aggregating text nodes into data fields is given in Algorithm 1.

To be statistically robust, the page-level relevance scores of text nodes can be aggregated to data fields. Specially, the attribute relevance vector of a data field D is computed by averaging over its member text nodes, as

$$(Rel_{DF}(D\,|\,a_1), \ldots, Rel_{DF}(D\,|\,a_m)),$$

where $Rel_{DF}(D \mid a_j) = \frac{1}{|D|} \sum_{t \in D} Rel_{TN}(t \mid a_j)$. Based on such relevance scores, each attribute a_j can be directly associated with the data field D_{a_j}

$$D_{a_j} = \arg\max_{D \in \mathbb{D}} Rel_{DF}(D\,|\,a_j).$$

The assembly

$$\mathbb{S}_{\mathcal{A}} = \{D_{a_1}, \ldots, D_{a_m}\}$$

is called a *solution* to the target attributes $\mathcal{A}$ on the new website. In practical, a threshold is set to remove data fields with low relevance scores in $\mathcal{S}_{\mathcal{A}}$, since some attributes may be unavailable in the new site.

6.3 Inter-Attribute Re-ranking

Aggregation to site-level data fields is noise-insensitive and more robust than page-level results. However, there are still difficult cases such as identifying `publish-date` from all date-type data fields in a book-seller site. As mentioned in Section 5.2, attribute layout is a good complement to semantics in such cases. For instance, the date-type data field being most close to `book-publisher` is likely to be the real `publish-date`. Therefore, it is natural to exploit characteristics of attribute layout to re-rank those possible solutions.

To generate more than one possible solutions, we need keep several candidate data fields for each attribute a_j. Specifically, data fields are first sorted in descending order according to $Rel_{DF}(D \mid a_j)$. Then the sorted list is truncated at the maximum drop point of the relevance score, yielding top-ranked data fields as candidates for a_j. Then, we enumerate every possible combination of attributes' data fields as a candidate solution to attributes $\mathcal{A}$.

From the perspective of semantics, the confidence of a candidate solution $\mathbb{S}^*_{\mathcal{A}}$ is defined as

$$Conf_{semantics}(\mathbb{S}^*_{\mathcal{A}}) = \sum_{j=1}^{m} Rel_{DF}(D_{a_j}\,|\,a_j), D_{a_j} \in \mathbb{S}^*_{\mathcal{A}}.$$

Algorithm 1 Aggregate-TextNodes-to-DataFields. Given a set of pages $\mathcal{P}_{sample}$, returns $\mathbb{D}$ consisting of all possible data fields in $\mathcal{P}_{sample}$.

```
1:  $\mathbb{D} = \emptyset$
2:  Initialize an empty hash-table $\mathcal{H}$
3:  for all page $p \in \mathcal{P}_{sample}$ do
4:    for all text node $t$ in $p$ do
5:      $h = \text{GenHashValue}(dom(t), preceding(t), prefix(t), suffix(t))$
6:      if $\mathcal{H}$ contains the key $h$ then
7:        Get the data field $D$ associated with $h$ from $\mathcal{H}$
8:        Add $t$ to $D$
9:      else
10:       Create a new data field $D = \{t\}$
11:       Add to $\mathcal{H}$ the new key $h$ associated with $D$
12:     end if
13:   end for
14: end for
15: for all $\langle h, D \rangle$ pair in $\mathcal{H}$ do
16:   if $|D| \geq 0.5|\mathcal{P}_{sample}|$ then
17:     Add $D$ to $\mathbb{D}$
18:   end if
19: end for
20: return $\mathbb{D}$
```

To measure the confidence from the perspective of inter-attribute layout, we first compute the layout matrix $\mathbf{A}^*$ for $\mathbb{S}^*_{\mathcal{A}}$, following the algorithm in Section 5.2. Then, analogous to cosine similarity, the layout confidence of $\mathbb{S}^*_{\mathcal{A}}$ is defined as

$$Conf_{layout}(\mathbb{S}^*_{\mathcal{A}}) = \frac{\langle \mathbf{A}^*, \mathbf{A} \rangle}{\|\mathbf{A}^*\|_F, \|\mathbf{A}\|_F},$$

where the numerator denotes element-wise inner product, and $\|\cdot\|_F$ is Frobenius norm. Finally, the optimal solution $\mathbb{S}_{\mathcal{A}}$ is the one with highest overall confidence, as

$$\mathbb{S}_{\mathcal{A}} = \arg\max_{\mathbb{S}^*_{\mathcal{A}}}(Conf_{semantics}(\mathbb{S}^*_{\mathcal{A}}) \times Conf_{layout}(\mathbb{S}^*_{\mathcal{A}})),$$

where multiplication is adopted instead of addition because the layout confidence is essentially designed for punishment.

7. EVALUATION AND DISCUSSION

In this section, we report the evaluation results of the proposed solution on real-world structured data extraction tasks. Extensive experiments were performed on a variety of websites and verticals to verify the performance of both the component algorithms and the overall system.

7.1 Experimental Settings

7.1.1 Dataset

To evaluate both the effectiveness and flexibility of our solution, we constructed a dataset consisting of around 124K pages collected from 80 websites. These websites are related to 8 semantically diverse verticals, including `Autos`, `Books`, `Cameras`, `Jobs`, `Movies`, `NBA Players`, `Restaurants`, and `Universities`. For each vertical, 10 popular websites were identified by issuing queries to search engines and mining the search results. For each website, $200 \sim 2,000$ pages, each containing structured data of one entity, were downloaded following the deep crawl technology in [13], given a few manually selected pages as examples of crawling targets.

For each vertical, a set of $(3 \sim 5)$ common attributes were selected as the targets of structured data extraction. To label the ground-truth of each website, a few handcrafted regular expressions were carefully prepared, to tag corresponding attribute values on each page. An overview of the dataset is given in Table 1[4].

Table 1: Overview of the experimental dataset.

Vertical	#Sites	#Pages	Attributes
`Autos`	10	17,923	`model, price, engine, fuel-economy`
`Books`	10	20,000	`title, author, ISBN-13, publisher, publish-date`
`Cameras`	10	5,258	`model, price, manufacturer`
`Jobs`	10	20,000	`title, company, location, date`
`Movies`	10	20,000	`title, director, genre, rating`
`NBA Players`	10	4,405	`name, team, height, weight`
`Restaurants`	10	20,000	`name, address, phone, cuisine`
`Universities`	10	16,705	`name, phone, website, type`

7.1.2 Performance Metrics

For vertical knowledge learning, all the pages in a seed site were involved; when adapting to another website, 50% pages from the new site were randomly selected for the adaptation. For testing performance, the evaluations were based on all the pages except for those from the seed site.

Precision and recall, as well as the F-score, were adopted as performance metrics in the evaluation. For each attribute, *precision* of a method is the number of pages whose ground-truth attribute values are correctly extracted, called page hits, divided by the number of pages from which the method extracts values; while *recall* is the page hits divided by the number of pages containing ground-truth attribute values. F-score is the harmonic mean of precision and recall. As a side note, it is possible that a page contains more than one ground-truth values of an attribute (*e.g.*, co-authors of a book); while the current solution in this paper is designed to detect only one attribute value. For this case, an extracted value is considered to be correct if it matches any labeled value in the ground-truth.

7.2 From One Seed Site to a Vertical

First, the proposed solution was evaluated under the ambitious scenario "from one tree to a forest". That is to say, for each vertical, only one website was leveraged as seed for knowledge learning. To demonstrate the contributions of various component algorithms in the solution, we implemented three versions of the proposed system, as:

– **PL (page-level semantic prediction)** is the version which just outputs the page-level attribute values predicted following Section 6.1. In other words, only the weak vertical knowledge learnt from the seed site is utilized.
– **PL+IP (inter-page aggregation)** is the version which aggregates page-level results into data fields, to correct page-level results through site-level voting. This is actually the output of Section 6.2.
– **PL+IP+IA (inter-attribute re-ranking)** is the full version of the proposed solution which further re-ranks the possible solutions according to the layout distributions of attributes, following the details in Section 6.3.

In addition, we also implemented a state-of-the-art work, the **Stacked Skews Model** (SSM) [5], whose target scenario is similar to ours. Moreover, the **SSM** approach also leverages site-level information, through aggregating features

[4]The dataset with ground-truth information is publicly available at http://swde.codeplex.com/.

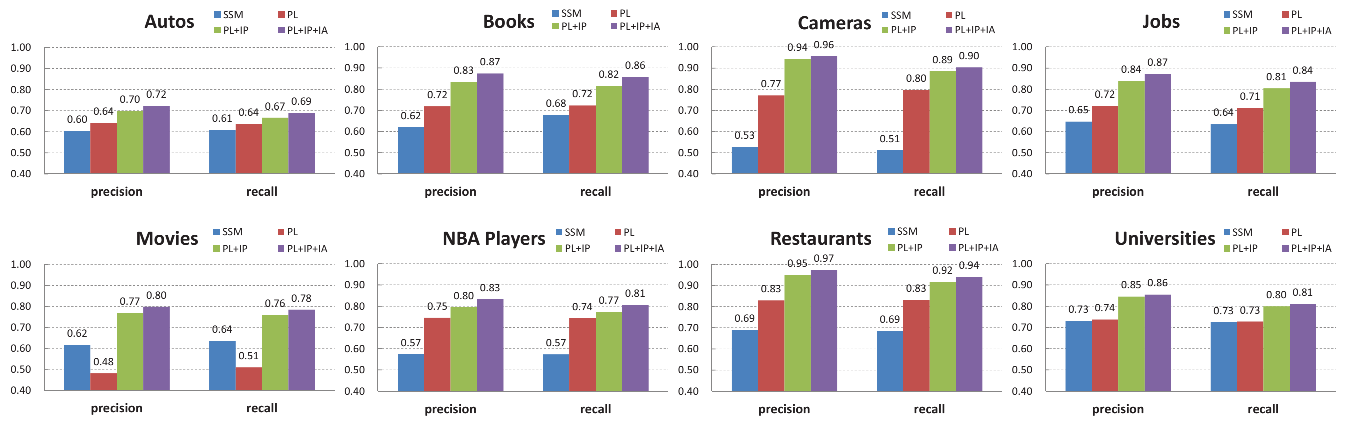


Figure 8: Comparisons of vertical-level performance of the stacked skews model (SSM) [5], PL (page-level semantic prediction), PL + IP (inter-page aggregation), and PL+IP+IA (inter-attribute re-ranking). The performance of each vertical is averaged over all its attributes and websites. The vertical knowledge was learnt based on only one seed site per vertical.

across a website. By contrast, our approach aggregates page-level extraction results rather than raw features.

For each vertical, all its 10 sites served as the seed site in turns to perform structured data extraction from the other 9 sites, yielding 90 runs of extraction in total. The vertical-level performance of each method was measured as the precision and recall averaged across all runs. Fig. 8 shows the overall results by vertical for each method, averaged across all the attributes in each vertical.

From Fig. 8, there are several observations. First, site-level information consistently and significantly boost the page-level results, for all the 8 verticals. Especially for the verticals `Cameras` and `Restaurants`, our solution can almost correctly identify values for all the attributes. The performance even exceeds our expectation as only one seed site was adopted for knowledge learning. Second, our performance is superior to that of the **SSM** approach, even the simplest **PL** version achieved better performance than **SSM** on 7 verticals. This is because **SSM** is designed for rich training data from multiple websites and is prone to overfit when training set is small (In Section 7.3 we will show the performance of **SSM** based on larger training set). Another reason is that the site-level feature aggregation is easy to be disturbed by misalignment, since precise tree-alignment is still an unsolved problem, as discussed in Section 6.2.

To show detailed evaluation results, we list the attribute-level performance of our system (**PL+IP+IA**) in Table 2. Although most results are very encouraging, there are still some attributes with unsatisfied performance. We analyzed some failure cases to diagnose the limitations of the current solution. For `height` of `NBA Players`, the main reason is its content format varies a lot (*e.g.*, 1.96m, 6 ft 5 in, 6'5", 6-5) across websites. The system has to rely on context features which are not always available. The `model` of `Autos` failed as there is no standard definition for this attribute. Different websites deliver different information for the `model` attribute. For instance, there are manufacturer, year, and even engine displacement listed under this attribute. The failure of `title` of `Movies` is due to multiple instances on a page. A page of a movie also contains other movies as "recommendation". Such recommendations tend to mislead the system away from the main entity of a page.

Table 2: Attribute-level performance of the proposed solution based on single seed site (averaged over different seed sites, in the form of "mean ± standard deviation").

Vertical	Attribute	Precision	Recall	F-score
Autos	model	0.46 ± 0.27	0.41 ± 0.26	0.43 ± 0.26
	price	0.80 ± 0.19	0.79 ± 0.19	0.80 ± 0.19
	engine	0.82 ± 0.14	0.82 ± 0.14	0.82 ± 0.14
	fuel-economy	0.81 ± 0.20	0.73 ± 0.18	0.77 ± 0.19
Books	title	0.89 ± 0.13	0.87 ± 0.14	0.88 ± 0.14
	author	0.95 ± 0.04	0.89 ± 0.04	0.92 ± 0.04
	ISBN-13	0.84 ± 0.19	0.84 ± 0.18	0.84 ± 0.18
	publisher	0.81 ± 0.06	0.81 ± 0.06	0.81 ± 0.06
	publish-date	0.88 ± 0.08	0.88 ± 0.08	0.88 ± 0.08
Cameras	model	0.93 ± 0.07	0.88 ± 0.06	0.90 ± 0.07
	price	0.98 ± 0.04	0.90 ± 0.05	0.94 ± 0.05
	manufacturer	0.96 ± 0.06	0.93 ± 0.06	0.94 ± 0.06
Jobs	title	0.99 ± 0.03	0.93 ± 0.04	0.95 ± 0.04
	company	0.84 ± 0.24	0.80 ± 0.22	0.82 ± 0.22
	location	0.87 ± 0.07	0.84 ± 0.07	0.85 ± 0.07
	date	0.79 ± 0.20	0.77 ± 0.19	0.78 ± 0.20
Movies	title	0.71 ± 0.25	0.68 ± 0.25	0.69 ± 0.25
	director	0.75 ± 0.11	0.80 ± 0.12	0.77 ± 0.12
	genre	0.96 ± 0.04	0.91 ± 0.04	0.93 ± 0.04
	rating	0.78 ± 0.23	0.75 ± 0.23	0.76 ± 0.23
NBA Players	name	0.84 ± 0.24	0.82 ± 0.23	0.83 ± 0.23
	team	0.82 ± 0.09	0.82 ± 0.09	0.82 ± 0.09
	height	0.76 ± 0.19	0.67 ± 0.17	0.71 ± 0.18
	weight	0.91 ± 0.10	0.91 ± 0.10	0.91 ± 0.10
Restaurants	name	0.95 ± 0.08	0.89 ± 0.07	0.92 ± 0.07
	address	0.97 ± 0.02	0.96 ± 0.02	0.96 ± 0.02
	phone	1.00 ± 0.00	0.98 ± 0.01	0.99 ± 0.00
	cuisine	0.98 ± 0.07	0.94 ± 0.06	0.96 ± 0.06
Universities	name	0.97 ± 0.05	0.95 ± 0.06	0.96 ± 0.06
	phone	0.79 ± 0.12	0.78 ± 0.12	0.79 ± 0.12
	website	0.96 ± 0.09	0.83 ± 0.08	0.89 ± 0.08
	type	0.70 ± 0.29	0.68 ± 0.27	0.69 ± 0.28

7.3 Multiple Seeds and Bootstrapping

The "one seed" scenario just proves that our solution has the ability to start with minimal human effort. Of course the solution can take multiple websites as seeds, as well as can accumulate vertical knowledge through bootstrapping. For bootstrapping, we simply keep those high-confidence extraction results of a website, and then consider the website as a new seed to the system. How to combine information from multiple seed sites is not a trivial task. Currently, we utilize a straightforward strategy without changing the sys-

Table 3: Average F-scores of the proposed solution based on multiple seed sites.

#Seeds	1	2	3	4	5
Our Solution	0.843	0.860	0.868	0.884	0.886
Our Solution (Bootstrap)	0.843	0.856	0.861	0.859	0.865
SSM	0.630	0.645	0.692	0.719	0.741

tem implementation too much. That is, an unseen website is evaluated based on each seed site independently, and the final solution is the one with the highest confidence score.

Table 3 shows the average F-scores of the proposed solution based on multiple seed sites. These seed sites can be manually labeled (*i.e.*, those in the ground-truth) or auto-tagged through bootstrapping. For comparison, we also trained the **SSM** model on multiple seeds manually labeled. From Table 3, we can see that the more websites involved in learning, the better the performance. When the seed sites were tagged through bootstrapping, the performance is a little worse than using manually labeled seeds. It is reasonable as the bootstrapping may introduce false examples into the learning process. Although the **SSM** method also benefits from more seed sites, its performance is still lower than the proposed solution. This again indicates the layout information (which was not fully adopted in **SSM**) is necessarily complementary to semantics to promote the performance.

8. CONCLUSIONS AND FUTURE WORK

In this paper, we propose a unified solution for extracting structured data from web sites based on one seed site given for a target vertical. The solution is general enough to tackle various verticals and relies on limited human effort to label one site for each vertical. A set of weak but general features are designed to characterize both semantic and layout knowledge of an arbitrary vertical. From a labeled seed site, we first extract features and learn vertical knowledge. Then, given a new site, the learnt knowledge is adapted to the site by exploiting site-level information, resulting in automatically identified attribute values for further wrapper induction. Experimental evaluations on 80 web sites in 8 diverse verticals showed promising results and demonstrated the feasibility and flexibility of the solution.

We consider several future directions to improve the proposed solution. First, a more principled strategy is desired to explicitly model and bridge the inter-site gaps in content and layout, with the number of observed sites increasing. Second, the solution could be combined with data record detection techniques to handle multiple entities in a page. From an experimental perspective, more extensive comparisons can be drawn to related approaches based on the published dataset. Moreover, it would be interesting to analyze the interactions among features, attributes, and seed sites, to get insight on feature design and seed site selection.

9. ACKNOWLEDGMENT

The first and third authors were partially supported by the National Natural Science Foundation of China (No. 60975001).

10. REFERENCES

[1] Document object model. http://en.wikipedia.org/wiki/Document_Object_Model.

[2] Freebase. http://www.freebase.com/.

[3] MSHTML reference. http://msdn.microsoft.com/en-us/library/aa741317(v=VS.85).aspx.

[4] A. Arasu and H. Garcia-Molina. Extracting structured data from web pages. In *Proc. SIGMOD*, pages 337–348, 2003.

[5] A. Carlson and C. Schafer. Bootstrapping information extraction from semi-structured web pages. In *Proc. ECML*, pages 195–210, 2008.

[6] C.-H. Chang and S.-C. Lui. Iepad: information extraction based on pattern discovery. In *Proc. WWW*, pages 681–688, 2001.

[7] V. Crescenzi, G. Mecca, and P. Merialdo. Roadrunner: Towards automatic data extraction from large web sites. In *Proc. VLDB*, pages 109–118, 2001.

[8] S. Dill, N. Eiron, D. Gibson, D. Gruhl, R. Guha, A. Jhingran, T. Kanungo, S. Rajagopalan, A. Tomkins, J. A. Tomlin, and J. Y. Zien. Semtag and seeker: bootstrapping the semantic web via automated semantic annotation. In *Proc. WWW*, pages 178–186, 2003.

[9] N. Kushmerick. *Wrapper induction for information extraction.* PhD thesis, 1997.

[10] B. Liu, R. Grossman, and Y. Zhai. Mining data records in web pages. In *Proc. KDD*, pages 601–606, 2003.

[11] I. Muslea, S. Minton, and C. Knoblock. A hierarchical approach to wrapper induction. In *Proc. AGENTS*, pages 190–197, 1999.

[12] S. Soderland. Learning information extraction rules for semi-structured and free text. *Mach. Learn.*, 34:233–272, 1999.

[13] M. L. A. Vidal, A. S. da Silva, E. S. de Moura, and J. M. B. Cavalcanti. Structure-driven crawler generation by example. In *Proc. SIGIR*, pages 292–299, 2006.

[14] T.-L. Wong and W. Lam. Learning to adapt web information extraction knowledge and discovering new attributes via a Bayesian approach. *TKDE*, 22:523–536, 2010.

[15] T.-L. Wong, W. Lam, and B. Chen. Mining employment market via text block detection and adaptive cross-domain information extraction. In *Proc. SIGIR*, pages 283–290, 2009.

[16] J.-M. Yang, R. Cai, Y. Wang, J. Zhu, L. Zhang, and W.-Y. Ma. Incorporating site-level knowledge to extract structured data from web forums. In *Proc. WWW*, pages 181–190, 2009.

[17] Y. Zhai and B. Liu. Web data extraction based on partial tree alignment. In *Proc. WWW*, pages 76–85, 2005.

[18] S. Zheng, R. Song, J.-R. Wen, and D. Wu. Joint optimization of wrapper generation and template detection. In *Proc. KDD*, pages 894–902, 2007.

[19] J. Zhu, Z. Nie, J.-R. Wen, B. Zhang, and W.-Y. Ma. 2d conditional random fields for web information extraction. In *Proc. ICML*, pages 1044–1051, 2005.

[20] J. Zhu, Z. Nie, J.-R. Wen, B. Zhang, and W.-Y. Ma. Simultaneous record detection and attribute labeling in web data extraction. In *Proc. KDD*, pages 494–503, 2006.