Learning Structural Co-occurrences for Structured Web Data Extraction in Low-Resource Settings

Zhenyu Zhang
Bowen Yu
Institute of Information Engineering, Chinese Academy of Sciences
School of Cyber Security, University of Chinese Academy of Sciences
Beijing, China
zhangzhenyu1996@iie.ac.cn
yubowen@iie.ac.cn

Tingwen Liu*
Tianyun Liu
Institute of Information Engineering, Chinese Academy of Sciences
School of Cyber Security, University of Chinese Academy of Sciences
Beijing, China
liutingwen@iie.ac.cn
liutianyun@iie.ac.cn

Yubin Wang
Li Guo
Institute of Information Engineering, Chinese Academy of Sciences
School of Cyber Security, University of Chinese Academy of Sciences
Beijing, China
wangyubin@iie.ac.cn
guoli@iie.ac.cn

ABSTRACT

Extracting structured information from all manner of webpages is an important problem with the potential to automate many real-world applications. Recent work has shown the effectiveness of leveraging DOM trees and pre-trained language models to describe and encode webpages. However, they typically optimize the model to learn the semantic co-occurrence of elements and labels in the same webpage, thus their effectiveness depends on sufficient labeled data, which is labor-intensive. In this paper, we further observe structural co-occurrences in different webpages of the same website: the same position in the DOM tree usually plays the same semantic role, and the DOM nodes in this position also share similar surface forms. Motivated by this, we propose a novel method, `Structor`, to effectively incorporate the structural co-occurrences over DOM tree and surface form into pre-trained language models. Such structural co-occurrences help the model learn the task better under low-resource settings, and we study two challenging experimental scenarios: website-level low-resource setting and webpage-level low-resource setting, to evaluate our approach. Extensive experiments on the public SWDE dataset show that `Structor` significantly outperforms the state-of-the-art models in both settings, and even achieves three times the performance of the strong baseline model in the case of extreme lack of training data.

CCS CONCEPTS

• **Information systems** → **Web mining**; **Data extraction and integration**.

KEYWORDS

web information extraction, structural co-occurrence, regular expression, low-resource setting

*Corresponding author.

WWW '23, April 30–May 04, 2023, Austin, TX, USA

ACM ISBN 978-1-4503-9416-1/23/04.
https://doi.org/10.1145/3543507.3583387

ACM Reference Format:
Zhenyu Zhang, Bowen Yu, Tingwen Liu, Tianyun Liu, Yubin Wang, and Li Guo. 2023. Learning Structural Co-occurrences for Structured Web Data Extraction in Low-Resource Settings. In *Proceedings of the ACM Web Conference 2023 (WWW '23), April 30–May 04, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3543507.3583387

1 INTRODUCTION

As the web is the largest knowledge base ever built by humans, web data extraction has gained the attention of researchers for more than twenty years [25, 41, 42], which in turn has various downstream applications such as knowledge-aware question answering [8, 34], information retrieval [2, 20], and recommendation systems [9, 29]. Typically, structured web data extraction systems focus on identifying attributes of interest for a topic entity from detail pages. A detail page denotes a webpage that presents a single data record, like a book in an online bookstore or a product in an e-commerce website [11, 21]. In contrast to text-intensive documents, such as essays, news reports, and legal contracts that are composed of grammatical sentences, text contents on webpages are often text fragments and not strictly grammatical. As a result, traditional natural language processing techniques are no longer directly applicable. Considering that the vast majority of web content is organized and exposed in HTML format, which also distinguishes the webpages from natural texts [4], it is a non-trivial task to extract structured data from webpages that are rendered with various HTML templates.

Indeed, the main goal of HTML is to support the logical presentation of web content, which brings new opportunities that are not available in natural texts. In particular, it is important to observe that detail pages from the same website usually share similar logical presentations and HTML templates. Take Figure 1 as an example, the right side shows two screenshots from Amazon.com, which are rendered to display in browser based on two source records and a shared DOM tree (a tree that describes the logical structure of webpages, as shown in the left side). The goal of structured web data extraction is to recognize attributes such as {`title`, `author`, `publisher`, `publication date`, `isbn13`} from these pages. An intuitive solution is to first distinguish webpages generated by different templates, and then learn an extractor for each template [5, 16, 39]. However, one major limitation is that they are supervised methods and require manual effort in preparing training examples for each template. Moreover, the learned extractor is not suitable for the

unseen website with unseen templates, thus considerable human efforts are required for either periodically updating templates or annotating unseen websites. In this work, we are interested in a scalable approach that is able to reduce expensive human efforts and extract attributes from unseen websites.

Modern neural networks learn from massive training data and achieve great generalization and robustness in many fields, while the subtext of limited labeled data is unsatisfactory performance [12]. In contrast, human beings have excellent transfer learning talent in low-resource scenarios, since humans learn the co-occurrences of data on the basis of understanding semantics and then abstract them into rules [37, 38]. When extracting attributes from webpages, it is easy to find that HTML elements playing the same semantic role in different webpages are often in the same position. For example, *"978-0590353403"* and *"978-0152023980"* in Figure 1 are both `isbn13` (International Standard Book Number), and located in the lower right corner. Another important finding is that elements in the same position also share similar surface, since when presenting specific semantic roles, the human consensus is to follow conventional expression patterns, especially for some date- and number-related fields. Back to the example above, we can roughly sum up that `isbn13` contains 13 digits, and there is a hyphen between the third and fourth digits. Now imagine that if a digital sequence matching this pattern also appears in the lower right corner of a webpage, we can infer that it is very likely to be `isbn13` of a book based on the two cases, even if we have never seen this sequence before. Such ability to summarize rules is the key for human beings to deal with low-resource scenarios better than neural networks. We name these two kinds of rules as *structural co-occurrences over DOM tree* and *surface form* respectively, because they go beyond the previous methods, which only optimize the semantic co-occurrences between elements and labels in a single web page, while we make more in-depth use of the shared cross-webpage DOM tree structure and element surface structure.

Motivated by the above observation, in this paper, we propose `Structor` to learn structural co-occurrences for structured web data extraction. We employ the pre-trained MarkupLM [19], a language model for webpages with a BERT-like architecture [7], as the base encoder to benefit from the pre-training process with rich and varied webpages. DOM nodes in a webpage are concatenated into a long sequence and sent into the encoder. To leverage the structural co-occurrence over DOM tree, for each DOM node, we retrieve a node in the same position from another DOM tree (i.e., another webpage) on the same website, and then incorporate it into the input sequence. To ensure that the inserted node does not interfere with the semantics of the original sequence, a semantic position embedding working in the embedding layer and a visible matrix working in the transformer layer are introduced. For the structural co-occurrence over surface form, we approximate the character patterns of DOM nodes by regular expressions and integrate them into the logits generated by neural networks. Although the context representation output by the encoder contains rich semantic information, we believe the surface forms containing many indicative signals are also the focuses of human attention.

We consider two challenging experimental scenarios in this paper, (i) website-level low-resource setting, where we learn a model

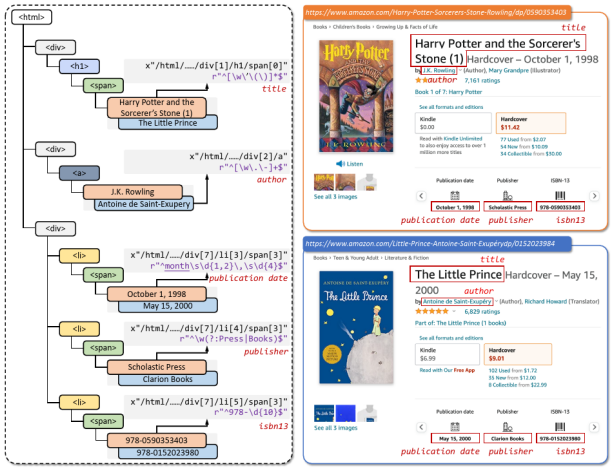


Figure 1: The schematic diagram of structural co-occurrence, where elements at the same position in one website play similar semantic roles and follow similar surface forms. The left part plots a DOM tree and the right displays two corresponding rendered webpages from Amazon.com.

with a few labeled seed websites and predict on other unseen websites from the same vertical, (ii) webpage-level low-resource setting, where we train a model with very limited labeled webpages from one website and do the same test process above. The first setting focuses on the transferability among websites and reduces the human efforts in scale, while the second assesses the effectiveness of the transferable model when lacking training data.

In summary, the main contributions of this paper include:

- To our best knowledge, it is the first work that efficiently exploits the structural co-occurrences over DOM tree and surface form for structured web data extraction.
- We are also the first to study the performance when reducing human efforts of data annotation in training stage.
- Extensive experiments on the public dataset, SWDE, show that `Structor` significantly outperforms the state-of-the-art methods, especially on the webpage-level setting.

Typesetting. In this paper, we use bold italic for the first appearance of a concept like ***DOM tree***, italic surrounded by quotations for words in the webpages like *"J.K. Rolling"*. Courier typeface surrounded by `x""` for XPaths like `x"/html/.../div[2]/a"` and `r""` for regular expressions like `r"^978-\d{10}$"`.

2 PRELIMINARIES

In this section, we first describe the structured web data extraction problem formally, and then introduce two kinds of structural co-occurrences in semi-structured webpages.

2.1 Problem Formulation

Here, we prefer to learn a transferable model for the structured web data extraction problem. Formally, each ***vertical*** (a.k.a. domain) v corresponds to a set of interesting ***attributes*** $\mathcal{A}_v$ (e.g., $\mathcal{A}_{book}$ = {`title`, `author`, `publisher`, `publication date`, `isbn13`}) and a

group of websites $\mathcal{W}_v$, where all the websites in a vertical share the same attribute set. Each website w_{vi} is composed of a collection of webpages which share a similar layout structure. Each webpage describes a single topic entity and can be parsed to a ***DOM tree***. The extraction task is typically formulated as *node classification* on the DOM tree [11]. Look at the right part of Figure 1, each element we can view in the browser corresponds to a ***DOM node***. Note that a DOM tree contains a ***variable node*** set, a fixed node set, and a non-text node set. Fixed nodes remain the same across different detail pages on the same website while variable nodes may contain different text contents. Following the convention in structured web data extraction [11], we assume a node corresponds to at most one attribute-type, and *narrow down the search range to variable nodes* because the attribute values should vary in different pages.

2.1.1 Website-Level Low-Resource Setting. Given *a small set of annotated seed websites* $\{w^s_{v1}, w^s_{v2}, ..., w^s_{vi}\}$ for a vertical v, we aim to learn a transferable model $\mathcal{M}$ to extract attributes from *a lager set of unseen websites* $\{w^u_{v1}, w^u_{v2}, ..., w^u_{vj}\}$ in the same vertical. It focuses on scalability in practical applications and is also the conventional setting in structured web data extraction [11, 21].

2.1.2 Webpage-Level Low-Resource Setting. In this scenario, we hope to further reduce the human cost, because the website-level setting still requires a lot of manpower to collect, clean, and label thousands of webpages. To reduce the labeling cost in the training stage, we leverage *very limited annotated webpages (less than 100) from only one website* to learn $\mathcal{M}$ and extract attributes from unseen websites in the same vertical. It is abstracted from the real scenario that a task needs to be annotated from scratch.

2.2 Structural Co-occurrences

Before introducing structural co-occurrences in webpages, we first clarify several related concepts. Unlike plain text with natural language, HTML (Hyper-Text Markup-Language) is the standard language for documents designed to be displayed in web browsers, and ***DOM*** (Document Object Model) tree is the tree structure object of markup-language-based documents (e.g., webpages) wherein each DOM node corresponds to an HTML element. ***XPath*** (XML Path Language) is a query language for selecting elements from markup-language-based documents and can be used to uniquely locate a DOM node based on the DOM tree. In a typical XPath expression, like `"/html/body/div[7]/li[5]/span[0]"`, the texts stand for the tag names while the subscripts are the ordinals when multiple nodes have the same tag name under a common parent node. We show an example of the DOM tree along with XPaths in Figure 1, from which we can identify the genealogy of all nodes within the document, as well as their XPath expressions.

2.2.1 Structural Co-occurrence over DOM Tree. In Figure 1, one obvious phenomenon is that webpages on the same website usually share a similar logical structure. The underlying principle is that in the process of website development, webpages are generated by filling different records in the same HTML template. Therefore, these webpages also share a similar DOM tree, and *DOM nodes with the same XPath play similar semantic roles.* Specifically, *"J.K. Rolling"* and *"Antoine de Saint-Exupery"* share the same XPath `x"/html/.../div[2]/a"`, so if we already know that *"J.K. Rolling"* is the `author` of *"Harry Potter"*[1], we can easily infer that *"Antoine de Saint-Exupery"* in the same position is the `author` of *"The Little Prince"*, even if we may not have any knowledge about him before. In other words, when predicting the attribute-type of *"Antoine de Saint-Exupery"*, it is feasible to leverage the relevant information of *"J.K. Rolling"*. We name such nodes that share the same XPath in different webpages as ***Peer*** nodes.

2.2.2 Structural Co-occurrence over Surface Form. When looking from DOM tree to DOM node, it is not difficult to find that *DOM nodes with the same semantic roles also have similar surface forms*, especially for dates and numbers. For example, *"October 1, 1998"* and *"May 15, 2000"* in Figure 1 are `publication date` for their topic entities, *"978-0590353403"* and *"978-0152023980"* are `isbn13`. They all follow some underlying character patterns, which are also the convention when people express specific types of knowledge and could be well approximated by ***RegExs*** (Regular Expressions) [35]. Specifically, `r/^month\s\d{1,2}\,\s\d{4}$/`[2] describes the date-type string and `isbn13` follows the `r/^\d{3}-\d{10}$/` pattern. This kind of knowledge is shared by all websites in the vertical, within and throughout all the webpages.

3 MODEL ARCHITECTURE

This section lays out `Structor`, a transformer-based model with structural co-occurrences, and introduces the details of each module. Firstly, we expound on our main motivation:

Motivation. Inspired by the great successes of pre-training/fine-tuning paradigm in low-resource NLP tasks [17, 30], we employ pre-trained MarkupLM [19] as the base encoder of webpages. To make full use of the structural co-occurrence over DOM tree, we first retrieve peer nodes with the same XPath for each variable node and splice them into input sequence. Then, we replace the default physical position embedding in input layer as semantic position embedding, and introduce a visible matrix to control the information propagation in transformer layer. In this way, the spliced peer nodes from other webpages are only associated with corresponding variable nodes, and do not affect the semantics of other tokens in the input sequence. Considering that human beings fully consider the character pattern of a string when understanding its semantic roles, we integrate the structural co-occurrence over surface form in the prediction stage. We collect and develop a series of regular expressions for each attribute type, and hypothesize that matching regular expressions is a necessary condition for the final prediction. In other words, only when a DOM node matches the specific regular expressions can it be corresponding attribute type. Throughout the whole process, there are no new parameters that need to be optimized from scratch, so the proposed approach is theoretically suitable for scenarios with limited training data.

3.1 Webpage Encoder with XPath

The webpage encoder is based on MarkupLM [19], a transformer-based model with XPath embedding, and also includes three key adaptations to integrate structural co-occurrence over DOM tree:

[1]We abbreviate *"Harry Potter and the Sorcerer's Stone (1)"* as *"Harry Potter"* for brevity.
[2]We follow the standard RegEx grammar (https://en.wikipedia.org/wiki/Regular_expression) and use "month" to denote twelve months (January, February, ...).

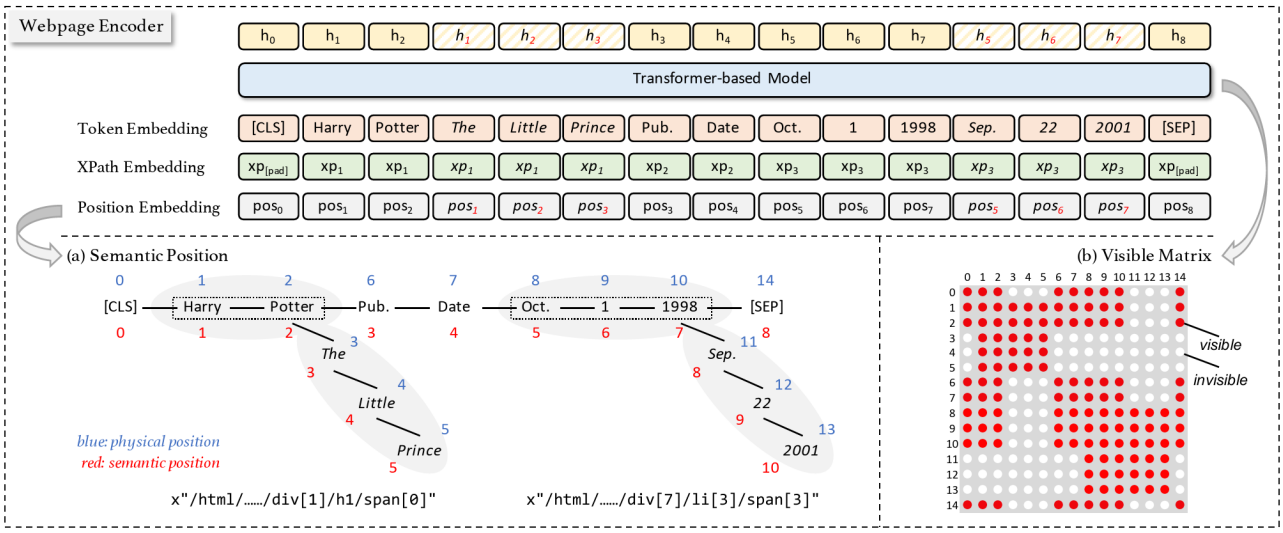


Figure 2: The architecture of webpage encoder. Variable nodes in the input sequence are extended by peer nodes. We leverage semantic position and visible matrix to make the introduced peer nodes only act on corresponding variable nodes. The semantic position (red number) ensures that the extension does not change the semantics of original input sequence. The visible matrix is multiplied by the attention weight matrix to realize the blocking and interaction of information.

i.e., input sequence with peer node, embedding layer with semantic position, and transformer layer with visible matrix. Figure 2 shows the architecture of webpage encoder.

3.1.1 Base Encoder with XPath. To take advantage of existing pre-trained models and adapt to markup-language-based webpage tasks, MarkupLM [19] follows the BERT [7] architecture and introduce XPath embedding to the original embedding layer.

For the i-th input token x_i, the XPath expression is split by "/", so as to obtain unit information at each level $[(t_0^i, s_0^i), ...(t_{n_u}^i, s_{n_u}^i)]$, where n_u is the max depth of XPath and (t_j^i, s_j^i) denotes the tag name and subscript of the XPath unit on level j. To convert XPath expression into XPath embedding, t_j^i and s_j^i are input into the j-th tag embedding layer and subscript embedding layer, respectively, and added up to get the j-th unit embedding u_j^i,

$$u_j^i = TagEmb_j(t_j^i) + SubEmb_j(s_j^i). \tag{1}$$

Next, all the unit embeddings are concatenated and fed into a feed-forward layer to get the final XPath embedding xp_i,

$$xp_i = W_p \left(\left[u_0^i; u_1^i; ...; u_{n_u}^i \right] \right) + b_p, \tag{2}$$

where $W_p \in \mathbb{R}^{d_h \times n_u d_u}$ and $b_p \in \mathbb{R}^{d_h}$ denotes the weight and bias of linear transformation, d_u and d_h denotes the dimension of unit embedding and XPath embedding, respectively.

For more details about XPath embedding and pre-training process, we recommend readers refer to the original paper [19], The next three adaptations are exactly the focus of our webpage encoder.

3.1.2 Input Sequence with Peer Node. There are usually thousands of nodes in one webpage, and some nodes are definitely not of interest to us. An intuitive idea is to only focus on variable nodes in the webpage instead of sending all nodes into the model, to avoid meaningless computational overhead, but it does not mean that the fixed nodes are useless for the problem. Take the *"Harry Potter"* webpage as example again, the preceding node *"Publication Date"* of *"October 1, 1998"* even directly indicates its attribute type. Therefore, when converting a webpage into an input sequence, we reserve a certain number of preceding nodes (either fix or variable node) before each variable node to enrich the overall semantics. It is also a conventional preprocessing when solving structured web data extraction in the era of deep learning [19, 21, 40].

Inspired by the process of enabling BERT representation with knowledge graph [22], we enhance the representation of variable nodes by integrating their peer nodes. For each variable node, we collect a peer node set $\mathcal{N}_p$ to store nodes from all webpages with the same XPath. Note that for any two nodes n_i and n_j in $\mathcal{N}_p$, n_i is the peer node of n_j, and vice versa. Next, we randomly select n_p peer nodes from $\mathcal{N}_p$ ($n_p = 1$ in default), then paste it behind the variable node. Finally, the sequence sent into the transformer-based model is as follows,

$$[CLS], ..., Tok_i^{prec}, Tok_i, Tok_i^{peer}, ..., [SEP], \tag{3}$$

where $Tok_i^{prec}, Tok_i, Tok_i^{peer}$, denote the tokens of preceding node, variable node and peer node, respectively.

3.1.3 Embedding Layer with Semantic Position. Similar to MarkupLM, we generate the input features by summing up token embedding, XPath embedding, and position embedding. For a transformer-based model, if there is no position embedding, it will be equivalent to a bag-of-word model, resulting in a lack of sequential information (i.e., the order of tokens). However, unlike common input sequences from the same document, the extended sequence with peer nodes contains a lot of information from other webpages, and the physical position is bound to cause information confusion. Therefore, how to distinguish inserted tokens while retaining the original sequential information is the key to position embedding.

All the sequential information of transformer-based models is contained in the position embedding, which allows us to introduce a semantic position to modify the physical position of tokens from peer nodes. Figure 2 shows a vivid example of the input sequence and semantic position, after attached tokens from peer nodes, *"The Little Prince"* are inserted after *"Harry Potter"*, but the book published on *"Oct 1 1998"* should be *"Harry Potter"* instead of *"The Little Prince"*. To solve this problem, we set the position number of *"Pub. Date"* to *3, 4* instead of *6, 7*. So when calculating attention score in the transformer, *"Pub."* is at the next position of *"Potter"* by the equivalent. However, another problem arises, the index of *"Pub."* and *"The"* are both *3*, which makes them close in position when calculating self-attention, but in fact, there is no connection between them. The solution to this problem is to introduce a visible matrix in the transformer layer, which is covered next.

3.1.4 Transformer Layer with Visible Matrix. The extended sequence contains knowledge from peer nodes, which helps the model understand the semantic role of input tokens. However, the risk raised with peer nodes is that it can lead to changes in the meaning of original sequence. For example in Figure 2, *"The Little Prince"* is only related to *"Harry Potter"* and has nothing to do with *"Sep. 22 2001"*, so the representation of *"Oct. 1 1998"* should not be affected by *"The Little Prince"*. Besides, the *"[CLS]"* token summarizing the whole sequence should not bypass *"Harry Potter"* to get the information of *"The Little Prince"*, as this would bring semantic changes. To prevent the risk from happening, we introduce a visible matrix $M \in \mathbb{R}^{n_s \times n_s}$ (n_s is the length of input sequence) to limit the visible area of each token so that *"The Little Prince"* and *"Sep. 22 2001"*, *"[CLS]"* and *"The Little Prince"* are not visible to each other,

$$M_{ij} = \begin{cases} 0, & t_i \Leftrightarrow t_j, \\ -\infty, & t_i \nLeftrightarrow t_j, \end{cases} \tag{4}$$

where $t_i \Leftrightarrow t_j$ indicates that t_i and t_j are tokens from the original input sequence or two peer nodes (i.e. visible to each other), while $t_i \nLeftrightarrow t_j$ are not, i and j are physical position indexes.

Typical transformer-based models (e.g., BERT, MarkupLM) leverage a fully-connected attention map to measure pairwise interaction between tokens. To prevent the false semantic changes, a straightforward operation is to modify the attention map by M,

$$Q^{l+1}, K^{l+1}, V^{l+1} = H^l W_q, H^l W_k, H^l W_v, \tag{5}$$

$$H^{l+1} = \text{softmax}\left(\frac{Q^{l+1} {K^{l+1}}^\top + M}{\sqrt{d_k}}\right) V^{l+1}, \tag{6}$$

where W_Q, W_K, and W_V are trainable parameters with shape $\mathbb{R}^{d_h \times d_h}$. H^l is the hidden states of the l-th transformer layer. d_k is the scaling factor. Intuitively, if t_i is invisible to t_j, M_{ij} masks the attention score to 0, which means t_i makes no contribution to t_j.

3.2 Attribute Predictor with RegEx

The webpage encoder generates a context representations for each input token. Instead of directly input them into the classifier, we introduce how to predict the attribute type for each variable node with structural co-occurrence over surface form in this section.

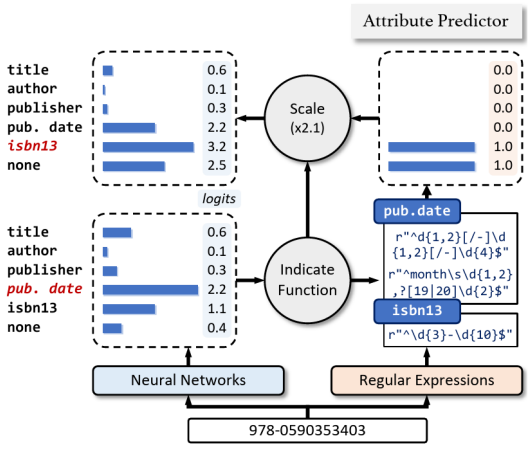


Figure 3: The overview of attribute predictor. The neural networks model the semantic of input node while the regular expressions capture surface forms. We show representative RegExs for `publication date` and `isbn13`, in which the input node only matches `isbn13`, so as to fix the prediction.

3.2.1 RegEx Collection. Regular expression is an algebraic notation used to describe specific patterns we want to match, which search through the sequence and return all texts meeting the pattern. When taking the extraction task as an attribute classification problem, regular expressions facilitate human experts to encode world knowledge about specific attribute types, especially for date- and number-related attributes. Specifically, we first collect regular expressions from *RegEx Library*[3], then summarize and supplement them manually, and finally get *at least one and at most five* expressions for each attribute. The collection process follows two principles, pattern matching and keyword matching. Pattern matching requires the sequence to strictly match the specific pattern, mainly for attributes with fixed form (e.g., date, phone) or without strict regularity (e.g., name, company). By comparison, keyword matching is hoped that the hit of keywords brings some positive signals to type judgment. For example, *"press"* and *"book"* are some iconic keywords for `publisher`.

3.2.2 Classification Layer with RegEx. With hidden states output by the last layer of transformers, neural-based methods typically utilize a classification layer with softmax to obtain the probability of each attribute. For the i-th token, the predicted attribute distribution of neural networks is,

$$nlogit_i = W_n h_i^L + b_n, \tag{7}$$

$$p_i = \text{softmax}(nlogit_i), \tag{8}$$

$$y_i = \arg\max(p_i). \tag{9}$$

where $W_n \in \mathbb{R}^{d_h \times d_a}$ and $b \in \mathbb{R}^{d_a}$ are trainable parameters, d_a is the size of pre-defined attribute set. $nlogit_i$ is the *logit* generated by neural networks. Obviously, the process only fits training data at semantic level, neglecting the structural co-occurrence over surface form, which sometimes falls into factual errors, such as predicting a digit string *"978-0590353403"* into `publication date`.

[3]https://regexlib.com

To take the surface form into consideration, for each DOM node, we check whether it matches regular expressions one by one and record the result as a 0-1 vector $R_i \in \mathbb{R}^{n_r d_a}$, where $n_r = 5$ is the maximum of regular expression for each attribute, 0 represents mismatch and 1 represents match. Next, a 0-1 matrix $W_r \in \mathbb{R}^{n_r d_a \times d_a}$ is introduced to map the results R_i to corresponding attributes,

$$rlogit_i = W_r R_i \tag{10}$$

However, there are two obstacles to combining the *logits* generated by neural networks and regular expression. As shown in Figure 3, the first is that $nlogit_i$ predicted by neural networks and the 0-1 vector $rlogit_i$ are not on the same scale. Thus we propose to adaptively scale $rlogit_i$ in Equation 11, so that $rlogit_i$ could make effective corrections to the false predictions of $nlogit_i$,

$$rlogit_i \leftarrow (\max(nlogit_i) - \min(nlogit_i)) \cdot rlogit_i \tag{11}$$

The second obstacle is that regular expressions may not cover all cases, since data in real scenes are quite diverse and complex. Here we use an indicator function to measure the confidence of $nlogit_i$, and only use $rlogit_i$ to update the predictions with low confidence. Otherwise, we directly output the prediction of neural network,

$$\mathbb{I}_r(rlogit_i) = \begin{cases} rlogit_i, & E(p_i) > \delta_l \ \& \ E(p_i) > \delta_g, \\ 0, & \text{other}, \end{cases} \tag{12}$$

$$E(p_i) = -\sum_j p_j^i \log p_j^i, \tag{13}$$

where $E(p_i)$ is the entropy of original predicted distribution, and a large entropy means low confidence, δ_l is the local threshold and set as a hyper-parameter, δ_g is the global threshold and set as the average entropy of all predictions in the input sequence. In other words, if the entropy of a prediction is high enough locally and globally, we update the prediction with RegEx knowledge. Finally, the prediction process (Equation 8) is modified to,

$$p_i = \text{softmax}(nlogit_i + \mathbb{I}_r(rlogit_i)), \tag{14}$$

4 EXPERIMENTS

In this section, we firstly introduce the experimental dataset and implementation details. Then, a collection of baseline models are included to compare with our model under the website-level and webpage-level low-resource settings.

4.1 Experimental Setup

4.1.1 Dataset. We carry out the publicly accessible Structured Web Data Extraction (SWDE) dataset [11] for all evaluation tasks, which has 8 verticals and 10 websites for each vertical[4]. Since there is no official train-test split for SWDE, in the *website-level low-resource* experiments, we follow the conventional setting [19, 21, 40] to randomly select k seed websites as training data and use the remaining $10 - k$ websites as the test set. In the *webpage-level low-resource* experiments, only a few labeled webpages from *one* website can be accessed in the training stage, and the test stage is carried out on all pages of the remaining *nine* websites. Different with the website-level setting, we leverage k to denote the seed webpages used in the training stage, and the value of k is $[10, 20, 50, 100]$.

[4] see also Table 4 in Appendix for the statistics of SWDE.

Table 1: Comparing performance of six baseline methods to Structor with different numbers of seed websites. Each entry is the mean value on all 8 verticals and 10 permutations of seed websites, thus 80 experiments in total.

Model \ *#Seed Sites*	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$
SSM [3]	63.00	64.50	69.20	71.90	74.10
Render-Full [11]	84.30	86.00	86.80	88.40	88.60
FreeDOM-NL [21]	72.52	81.33	86.44	88.55	90.28
FreeDOM-Full [21]	82.32	86.36	90.49	91.29	92.56
LANTERN [40]	83.06	88.96	91.63	92.84	93.75
MarkupLM [19]	82.11	91.29	94.42	95.31	95.89
Structor	**84.48**	**92.12**	**95.03**	**95.79**	**96.08**

Obviously, the same website or webpage is never present in training and test data in any experiment.

4.1.2 Implementation Details. We implement Structor[5] with transformers [31] and the pre-trained *markuplm-base*[6] model. We follow the same data pre/post-processing, develop environment and hyper-parameter setting as in MarkupLM [19] for fair composition. Specially, we carry out the webpage-level low-resource experiments with only one NVIDIA V100 GPU to alleviate the data coefficient problem (when k is small, if multiple GPUs are used, the average data allocated to each GPU is much lower than the set batch size). Besides, the number of peer nodes for each variable node is set to 1 for efficiency reasons, the local threshold in δ_l Attribute Predictor is set to 0.1 (chosen from {0.01, 0.05, 0.1, 0.2}).

We evaluate predicted attribute values with the true values for each detail page and then compute the average F1 score over all websites/verticals. In both two settings, we take cyclic permutations after fixing an order within the websites for each vertical, thus the final result per vertical is obtained by taking the average of all 10 permutations of seed websites for each k.

4.1.3 Compared Baselines. In the website-level low-resource setting, we compare Structor against recent representative baselines[7], including Stacked Skews Model (SSM [3]), Relational Neural Model (Render [11]), Relational Neural Model (FreeDOM [21]), Simplified DOM Model (LANTERN [40]), and Pre-trained Language Model (MarkupLM [19]). In the webpage-level setting, we reimplement the latest open-source work MarkupLM as the strongest baseline, which is also the base encoder of our model. Another key component of Structor is RegEx, but a candidate field might match multiple regular expressions of multiple attributes, that is, the attribute cannot be determined, so RegEx cannot be treated as an independent evaluable baseline method. Recently, there are also some other works on SWDE, such as DOM-LM [6] and WebFormer [27], but they reorganized the data to focus on the performance under full training, and therefore is not considered in this paper.

4.2 Website-Level Low-Resource Setting

We first compare the performance of all baselines and Structor with different numbers of seed websites. Then, we conduct two sets of analyses to verify the effectiveness of our design choices.

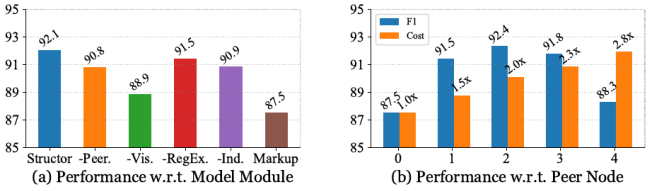
[5] https://github.com/zzysay/structor

[6] https://huggingface.co/microsoft/markuplm-base

[7] see also Appendix for detailed descriptions of baselines.

Table 2: The detailed performance of Structor of using different numbers of seed websites in different verticals. Each entry is the mean value of 10 permutations of seed websites.

Vertical \ #Seed Sites	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$
Auto	75.75	88.94	94.30	95.35	97.40
Book	79.99	86.80	90.23	90.66	90.21
Camera	90.24	93.71	95.67	96.92	97.03
Job	75.27	86.48	90.95	90.40	91.14
Movie	85.79	95.66	98.28	99.02	98.86
NBA Player	92.06	94.31	95.76	96.68	96.69
Restaurant	87.09	94.62	97.09	98.82	98.57
University	89.67	96.42	97.91	98.51	98.77
Average	84.48	92.12	95.03	95.79	96.08



(a) Performance w.r.t. Model Module (b) Performance w.r.t. Peer Node

Figure 4: Performance analysis about model modules and peer nodes. Best view in color. Here we take NBA Player, the vertical with the least amount of data, as an example and the remaining seven verticals share the similar tendency.

4.2.1 Main Results. In Table 1, we show the overall comparison between our Structor and baselines using different numbers of seed websites. Our model consistently outperforms the state-of-the-art strong baselines and obtains the largest performance lift compared with MarkupLM when k=1 (i.e., train on only 1 website and infer on the other 9 websites). We believe that when the amount of training data is relatively small, the two kinds of structural co-occurrences we introduced ensure the lower limit of performance. In contrast, when more data join the training process, MarkupLM is also able to learn the regular patterns from the data, which overlaps with our structural co-occurrences to some extent.

We also report the detailed performance of Structor in Table 2. The absolute improvement on F1 increases all the way to using 5 seed websites with gradually diminishing improvements. It is not surprising because more training data obtain better coverage of all possible instances, while the model becomes more robust and less new knowledge can be transferred from annotated websites to unseen websites when there is enough training data.

4.2.2 Ablation Study. To concretely examine the effectiveness of our design philosophy, we demonstrate ablation studies on different modules of Structor. As shown in Figure 4(a), both structural co-occurrence over DOM tree and surface form (i.e., -Peer./ -RegEx) make important contributions to the final performance. We conclude that the peer nodes provide strong auxiliary information for attribute prediction, and the combination of regular expressions and neural networks allows us to exploit the conciseness and effectiveness of regular expressions and the strong generalization ability of neural networks. Apart from that, we also test the importance of associated components when applying structural co-occurrences.

Table 3: Comparing performance of MarkupLM to Structor with different number of seed webpages. Each entry is the mean value of 5 independent sub-samples and 10 permutations, thus 50 experiments in total.

Vertical \ #Seed Pages		$k=10$	$k=20$	$k=50$	$k=100$
Auto	LANTERN	51.08	65.87	69.38	70.55
	MarkupLM	11.53	41.62	63.29	69.10
	Structor	52.41	61.42	68.86	70.23
Book	LANTERN	27.12	29.60	49.90	58.41
	MarkupLM	8.71	22.81	56.19	70.93
	Structor	50.10	60.40	69.75	71.56
Camera	LANTERN	67.61	62.88	76.00	81.20
	MarkupLM	30.38	65.42	79.49	82.58
	Structor	67.45	77.49	84.27	87.43
Job	LANTERN	52.84	56.95	71.99	73.03
	MarkupLM	17.76	25.00	55.24	69.23
	Structor	46.97	59.04	62.59	71.56
Movie	LANTERN	36.66	49.27	57.33	63.06
	MarkupLM	7.99	24.25	69.75	82.02
	Structor	44.84	58.33	76.26	83.55
NBA Player	LANTERN	39.55	42.55	49.81	62.02
	MarkupLM	14.23	34.31	80.10	85.97
	Structor	63.25	78.56	87.85	89.91
Restaurant	LANTERN	38.97	51.64	62.72	74.93
	MarkupLM	19.04	33.54	68.29	77.93
	Structor	66.64	72.06	84.32	85.11
University	LANTERN	38.18	52.13	67.91	72.04
	MarkupLM	23.91	39.63	66.79	77.77
	Structor	65.17	79.66	82.48	85.23
Average	LANTERN	44.01	51.36	63.13	69.40
	MarkupLM	16.69	35.82	67.39	76.94
	Structor	**57.10**	**68.37**	**77.04**	**80.57**

The results show that the semantic position and visible matrix (i.e., -Vis.) are very important because they maintain the structure and semantics of the input sequence. Similarly, the overall performance also decreases when ablating the indicator function (i.e., -Ind.), we hypothesize that arbitrarily fixing all the predictions degrades the neural network and makes it fall into local optimum.

4.2.3 Peer Node Number Analysis. Although peer node brings rich semantic information, it extends the input sequence, so that increases the overall computational cost. As shown in Figure 4(b), we conduct quantitative analysis and report the F1 scores and time costs with {0,1,2,3,4} peer nodes, in which the relative cost is reported based on the cost of not using peer nodes (i.e., 0 peer nodes). When integrating more and more peer nodes, the computational cost increases monotonously while the performance first increases and then decreases, because too many peer nodes might drown the semantics of input sequence, resulting in a negative impact. Besides, the performance lift brought by two peer nodes is relatively limited compared with one, Therefore, utilizing one peer node is a choice to balance performance and efficiency.

4.3 Webpage-Level Low-Resource Setting

In this part, we test MarkupLM and Structor by varying the number of seed webpages from {10, 20, 50, 100} to show the impact of our approach in a more demanding low-resource scenario, assuming a task that needs to be annotated from the scratch.

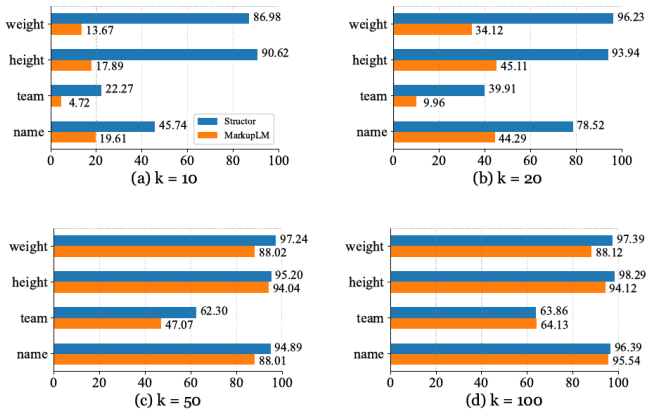


Figure 5: Per-attribute performance comparisons between `MarkupLM` and `Structor` in the `NBA Player` vertical with different number of seed webpages. Numerical attributes like height and weight achieve the largest performance lifts.

4.3.1 Main Results. Table 3 reports the detailed performance of `MarkupLM` and `Structor` with different number of seed webpages. The most striking finding is that the performance of `MarkupLM` almost collapsed when the amount of training data is particularly limited ($k \leq 10$), while our `Structor` brings a quite amazing improvement, even reaching three times the performance of the baseline model. We explain that in this case, the training data is not enough to adjust the model parameters, while the structural co-occurrence over DOM tree can be regarded as a data augmentation solution, and the structural co-occurrence over surface form helps correct a large number of false predictions. Moreover, Compared with the traditional methods of using about 2000 webpages (that is, k=1 in Table 2), `Structor` is also capable of achieving competitive performance with only 100 seed webpages. It provides more imagination for the practical application of the proposed method in real world low-resource scenarios since the method greatly reduces the demand of the model for training data.

4.3.2 Per-Attribute Analysis. To access details of the indelible performance improvement, we plot four histograms in Figure 5 to show the per-attribute performance comparison between `MarkupLM` and `Structor` in the `NBA Player` vertical. From the results, we conclude that `Structor` perform very well on all kinds of attributes, which means the proposed model learns gratifying semantic and structural co-occurrences. Furthermore, numerical attributes possess the most remarkable performance improvements, which we believe is a great success of regular expressions, because such numerical attributes usually have some fixed formats (e.g., height: 6'11", weight: 220 lbs.), and the philosophy of pattern matching when collecting regular expressions enables us to accurately capture possible attribute values.

5 RELATED WORK

Structured Web Data Extraction. Structured data extraction from web documents has drawn a lot of attention from the data mining research community [4, 32, 41, 42]. Traditional solutions [5, 10, 25] usually require a significant number of human-crafted rules or labels for inducing a wrapper, which is not scalable if we wish to extract information from numerous websites. By contrast, Carlson et al. [3] and Hao et al. [11] propose to learn transferable models for the extraction of unseen websites without using new human annotations, but these rendering-based methods require carefully crafted heuristics around visual proximity to work well with expensive features. Recent works [19, 21, 40] focus on introducing neural networks into web extraction and incorporating with DOM trees. However, they lack a holistic view like humans and a large amount of training data is necessary to achieve promising results. In this paper, we consider an ambitious scenario. That is, to train a general enough model with limited human efforts and handle various webpages in one vertical without re-implementation.

Language Model with External Knowledge. Many efforts are devoted to pre-trained language models for learning informative representations [7, 24, 26], some works also show that extra knowledge, such as facts in WikiData and WordNet, can further benefit the pre-trained models [18, 33, 36], but the embeddings of words in the text and entities in the knowledge base are not in the same space so that a cumbersome pre-training stage is required. Recently, K-BERT [22] proposes a knowledge-enabled language model to handle the heterogeneous embedding space problem, and K-Adapter [28] introduces an adapter layer and clamps the pre-trained parameters in the knowledge infusion process. One major difference is that we systematically consider structural co-occurrences from the corpus rather than rely on a knowledge graph. Another unique aspect of this work is that the homogeneous knowledge is utilized plug-and-play without any new parameters and pre-training process.

Neural Network with Regular Expression. Regular expressions complement the robustness of neural networks by providing control of a rule-based system [1, 13, 35]. Existing research studies have exploited the significance of regular expression in various natural language processing tasks, especially in the absence of enough training data [14, 15, 23]. However, they all give the collected regular expressions absolute confidence and incorporate them into neural networks without considering their accuracy. It is also our biggest difference from previous methods, that is, we doubt regular expressions and propose an entropy-based method to selectively update the prediction of neural network.

6 CONCLUSION

This paper explores a transferable manner of learning structural co-occurrences to reduce the human efforts of structured web data extraction. It is the first attempt to leverage cross-webpage knowledge to capture the relevance of elements from the perspective of logical position and surface form. This kind of meta information is able to be transferred across websites and can thus help solve the web data extraction problem. Extensive experiments show the effectiveness of the proposed method, especially when the training data is extremely scarce. In the future, world knowledge will be taken into account as long as a public knowledge graph of that verticals can be easily accessed, we also plan to develop a unified model that is able to adaptively accept new attributes and extract structured knowledge for all verticals without training from scratch.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their insightful comments and constructive suggestions. This work is supported by the National Key Research and Development Program of China (grant No.2021YFB3100600), the Strategic Priority Research Program of Chinese Academy of Sciences (grant No.XDC02040400) and the Youth Innovation Promotion Association of Chinese Academy of Sciences (grant No.2021153).

REFERENCES

[1] Waheed Ahmed Abro, Guilin Qi, Zafar Ali, Yansong Feng, and Muhammad Aamir. 2020. Multi-turn intent determination and slot filling with neural networks and regular expressions. *Knowledge-Based Systems (KBS)* 208 (2020), 106428.

[2] Maristella Agosti, Stefano Marchesin, and Gianmaria Silvello. 2020. Learning unsupervised knowledge-enhanced representations to reduce the semantic gap in information retrieval. *ACM Transactions on Information Systems (TOIS)* 38, 4 (2020), 1–48.

[3] Andrew Carlson and Charles Schafer. 2008. Bootstrapping information extraction from semi-structured web pages. In *Proceedings of the 2008 Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD)*. 195–210.

[4] Valerio Cetorelli, Paolo Atzeni, Valter Crescenzi, and Franco Milicchio. 2021. The smallest extraction problem. *Proceedings of the VLDB Endowment (VLDB)* 14, 11 (2021), 2445–2458.

[5] Nilesh Dalvi, Ravi Kumar, and Mohamed Soliman. 2010. Automatic Wrappers for Large Scale Web Extraction. *Proceedings of the VLDB Endowment (VLDB)* 4, 4 (2010), 219–230.

[6] Xiang Deng, Prashant Shiralkar, Colin Lockard, Binxuan Huang, and Huan Sun. 2022. DOM-LM: Learning Generalizable Representations for HTML Documents. *arXiv preprint arXiv:2201.10608* (2022).

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. 4171–4186.

[8] Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *Proceedings of the 20th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*. 1156–1165.

[9] Sandra Garcia Esparza, Michael P O'Mahony, and Barry Smyth. 2010. On the real-time web as a source of recommendation knowledge. In *Proceedings of the 4th ACM Conference on Recommender Systems (RecSys)*. 305–308.

[10] Pankaj Gulhane, Amit Madaan, Rupesh Mehta, Jeyashankher Ramamirtham, Rajeev Rastogi, Sandeep Satpal, Srinivasan H Sengamedu, Ashwin Tengli, and Charu Tiwari. 2011. Web-scale information extraction with vertex. In *Proceedings of the IEEE 27th International Conference on Data Engineering (ICDE)*. 1209–1220.

[11] Qiang Hao, Rui Cai, Yanwei Pang, and Lei Zhang. 2011. From one tree to a forest: a unified solution for structured web data extraction. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 775–784.

[12] Michael A Hedderich, Lukas Lange, Heike Adel, Jannik Strötgen, and Dietrich Klakow. 2021. A Survey on Recent Approaches for Natural Language Processing in Low-Resource Scenarios. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. 2545–2568.

[13] Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. 2016. Harnessing Deep Neural Networks with Logic Rules. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*. 2410–2420.

[14] Chengyue Jiang, Zijian Jin, and Kewei Tu. 2021. Neuralizing Regular Expressions for Slot Filling. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 9481–9498.

[15] Chengyue Jiang, Yinggong Zhao, Shanbo Chu, Libin Shen, and Kewei Tu. 2020. Cold-start and interpretability: Turning regular expressions into trainable recurrent neural networks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 3193–3207.

[16] Furkan Kocayusufoglu, Ying Sheng, Nguyen Vo, James Wendt, Qi Zhao, Sandeep Tata, and Marc Najork. 2019. Riser: Learning better representations for richly structured emails. In *Proceedings of the 2019 World Wide Web Conference (WWW)*. 886–895.

[17] Dong-Ho Lee, Mahak Agarwal, Akshen Kadakia, Jay Pujara, and Xiang Ren. 2022. Good Examples Make A Faster Learner: Simple Demonstration-based Learning for Low-resource NER. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*. 2687–2700.

[18] Yoav Levine, Barak Lenz, Or Dagan, Ori Ram, Dan Padnos, Or Sharir, Shai Shalev-Shwartz, Amnon Shashua, and Yoav Shoham. 2020. SenseBERT: Driving Some Sense into BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*. 4656–4667.

[19] Junlong Li, Yiheng Xu, Lei Cui, and Furu Wei. 2022. MarkupLM: Pre-training of Text and Markup Language for Visually-rich Document Understanding. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*. 6078–6087.

[20] Xiangsheng Li, Jiaxin Mao, Weizhi Ma, Yiqun Liu, Min Zhang, Shaoping Ma, Zhaowei Wang, and Xiuqiang He. 2021. Topic-enhanced knowledge-aware retrieval model for diverse relevance estimation. In *Proceedings of the Web Conference 2021 (WWW)*. 756–767.

[21] Bill Yuchen Lin, Ying Sheng, Nguyen Vo, and Sandeep Tata. 2020. FreeDOM: A Transferable Neural Architecture for Structured Information Extraction on Web Documents. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*. 1092–1102.

[22] Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. 2020. K-BERT: Enabling Language Representation with Knowledge Graph. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI)*. 2901–2908.

[23] Bingfeng Luo, Yansong Feng, Zheng Wang, Songfang Huang, Rui Yan, and Dongyan Zhao. 2018. Marrying Up Regular Expressions with Neural Networks: A Case Study for Spoken Language Understanding. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*. 2083–2093.

[24] Qiming Peng, Yinxu Pan, Wenjin Wang, Bin Luo, Zhenyu Zhang, Zhengjie Huang, Teng Hu, Weichong Yin, Yongfeng Chen, Yin Zhang, Shikun Feng, Yu Sun, Hua Wu, and Haifeng Wang. 2022. ERNIE-Layout: Layout Knowledge Enhanced Pre-training for Visually-rich Document Understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2022 (EMNLP: Findings)*. 3744–3756.

[25] Sandeep Tata, Navneet Potti, James B Wendt, Lauro Beltrão Costa, Marc Najork, and Beliz Gunel. 2021. Glean: structured extractions from templatic documents. *Proceedings of the VLDB Endowment (VLDB)* 14, 6 (2021), 997–1005.

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of 31st Conference on Neural Information Processing Systems (NeurIPS)*. 1–12.

[27] Qifan Wang, Yi Fang, Anirudh Ravula, Fuli Feng, Xiaojun Quan, and Dongfang Liu. 2022. WebFormer: The Web-page Transformer for Structure Information Extraction. In *Proceedings of the ACM Web Conference 2022 (WWW)*. 3124–3133.

[28] Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuan-Jing Huang, Jianshu Ji, Guihong Cao, Daxin Jiang, and Ming Zhou. 2021. K-Adapter: Infusing Knowledge into Pre-Trained Models with Adapters. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021 (ACL: Findings)*. 1405–1418.

[29] Xiang Wang, Yaokun Xu, Xiangnan He, Yixin Cao, Meng Wang, and Tat-Seng Chua. 2020. Reinforced negative sampling over knowledge graph for recommendation. In *Proceedings of the Web Conference 2020 (WWW)*. 99–109.

[30] Yaqing Wang, Haoda Chu, Chao Zhang, and Jing Gao. 2021. Learning from Language Description: Low-shot Named Entity Recognition via Decomposed Framework. In *Findings of the Association for Computational Linguistics: EMNLP 2021 (EMNLP: Findings)*. 1618–1630.

[31] Thomas Wolf, Julien Chaumond, Lysandre Debut, Victor Sanh, Clement Delangue, Anthony Moi, Pierric Cistac, Morgan Funtowicz, Joe Davison, Sam Shleifer, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations (EMNLP: Demo)*. 38–45.

[32] Tak-Lam Wong, Wai Lam, and Tik-Shun Wong. 2008. An unsupervised framework for extracting and normalizing product attributes from multiple web sites. In *Proceedings of the 31st International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 35–42.

[33] Wenhan Xiong, Jingfei Du, William Yang Wang, and Veselin Stoyanov. 2020. Pretrained Encyclopedia: Weakly Supervised Knowledge-Pretrained Language Model. In *Proceedings of the 8th International Conference on Learning Representations (ICLR)*. 1–13.

[34] Wenhan Xiong, Mo Yu, Shiyu Chang, Xiaoxiao Guo, and William Yang Wang. 2019. Improving Question Answering over Incomplete KBs with Knowledge-Aware Reader. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*. 4258–4264.

[35] Shanshan Zhang, Lihong He, Slobodan Vucetic, and Eduard Dragut. 2018. Regular expression guided entity mention mining from noisy web data. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 1991–2000.

[36] Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: Enhanced Language Representation with Informative Entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*. 1441–1451.

[37] Zhenyu Zhang, Xiaobo Shu, Bowen Yu, Tingwen Liu, Jiapeng Zhao, Quangang Li, and Li Guo. 2020. Distilling Knowledge from Well-informed Soft Labels for Neural Relation Extraction. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI)*. 9620–9627.

[38] Zhenyu Zhang, Bowen Yu, Xiaobo Shu, Xue Mengge, Tingwen Liu, and Li Guo. 2021. From What to Why: Improving Relation Extraction with Rationale Graph. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021 (ACL: Findings)*. 86–95.

[39] Shuyi Zheng, Ruihua Song, Ji-Rong Wen, and C Lee Giles. 2009. Efficient record-level wrapper induction. In *Proceedings of the 18th ACM Conference on Information and knowledge Management (CIKM)*. 47–56.

[40] Yichao Zhou, Ying Sheng, Nguyen Vo, Nick Edmonds, and Sandeep Tata. 2022. Learning Transferable Node Representations for Attribute Extraction from Web Documents. In *Proceedings of the 15th ACM International Conference on Web Search and Data Mining (WSDM)*. 1479–1487.

[41] Jun Zhu, Zaiqing Nie, Ji-Rong Wen, Bo Zhang, and Wei-Ying Ma. 2006. Simultaneous record detection and attribute labeling in web data extraction. In *Proceedings of the 12th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*. 494–503.

[42] Jun Zhu, Bo Zhang, Zaiqing Nie, Ji-Rong Wen, and Hsiao-Wuen Hon. 2007. Webpage understanding: an integrated approach. In *Proceedings of the 13th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*. 903–912.

APPENDIX

The detailed statistics of experimental dataset is list in Table 4. Each vertical specifies 3~5 fields to extract and can be regarded as an independent subset of data. Each website has hundreds of pages, where a page has about 300 variable nodes for the model to classify. Following previous work [19, 21, 40], we evaluate the extraction performance by webpage-level F1 scores, which is the harmonic mean of precision and recall in each webpage. we use website-level XPath voting to find the XPath selected as the field value by the majority pages and correct the rest of the pages to extract field value from this XPath as well. Finally, `Structor` is compared with the following baselines:

Stacked Skews Model (`SSM`*).* `SSM` [3] utilizes expensive handcrafted features and tree alignment algorithms to align the unseen webpages with seed webpages, which is the feature-based state-of-the-art method that did not require visual rendering features.

Rendering Feature Model (`Render`*).* `Render` [11] employs visual features to explore the distance between each block in the web browser rendered result. The visual distance is very helpful but the rendering requires downloading and executing a large amount of external scripts, images, and style files, which are extremely time/space-consuming. In specific, `Render-Full` equipped with a sophisticated heuristic algorithm to compute visual distances gives the best performance compared to other variants.

Relational Neural Model (`FreeDOM`*).* `FreeDOM` [21] leverages a relational neural network to encode features such as relative distance and text semantics, where the first stage (`FreeDOM-NL`) learns a dense representation for each DOM tree node, and the relational neural network in the second stage (`FreeDOM-Full`) captures the distance and semantic relatedness between pairs of nodes in the DOM trees. It is the first systematic neural network solution for the problem and does not rely on visual features, but the two-stage model is hard to be deployed in practice.

Simplified DOM Model (`LANTERN`*).* `LANTERN` [40] simplifies the DOM trees to extract informative and transferable knowledge by keeping all the basic HTML element tags while removing the formatting and style tags. It models the rich structural information in the DOM tree such as friend circles, and learns a rich representation for each DOM tree node without using any visual features. However, the focus stays inside the webpage, ignoring the high-level structural co-occurrences over DOM tree and surface form.

Table 4: The statistics of SWDE. #Pages denotes the total webpage number in a vertical, #Nodes denotes the average variable node number in a webpage.

Vertical	#Pages	#Nodes	Fields
`Auto`	17,923	130.1	model, price, engine, fuel_economy
`Book`	20,000	476.8	title, author, isbn, pub, date
`Camera`	5,258	351.8	model, price, manufacturer
`Job`	20,000	374.7	title, company, location, date_posted
`Movie`	20,000	284.6	title, director, genre, mpaa_rating
`NBA Player`	4,405	321.5	name, team, height, weight
`Restaurant`	20,000	267.4	name, address, phone, cuisine
`University`	16,705	186.2	name, phone, website, type

Pre-trained Language Model (`MarkupLM`*).* `MarkupLM` [19] uses the DOM tree in markup language and the XPath query language to obtain the markup streams along with natural texts in markup-language-based documents. Specifically, an XPath embedding layer and three pre-training strategies are proposed for the transformer-based model to accept markup sequence inputs. We incorporate `MarkupLM` into our `Structor` as base encoder, so it is treated as the most important comparison model in all the experiments.