WebFormer: The Web-page Transformer for Structure Information Extraction

Qifan Wang*
Facebook AI
Menlo Park, CA, USA
wqfcr@fb.com

Yi Fang
Santa Clara University
Santa Clara, CA, USA
yfang@scu.edu

Anirudh Ravula
Google Research
Mountain View, CA, USA
ravulaanirudh25@gmail.com

Fuli Feng
University of Science and Technology
Hefei, China
fulifeng93@gmail.com

Xiaojun Quan
Sun Yat-sen University
Guangzhou, China
quanxj3@mail.sysu.edu.cn

Dongfang Liu*
Rochester Institute of Technology
Rochester, NY, USA
dongfang.liu@rit.edu

ABSTRACT

Structure information extraction refers to the task of extracting structured text fields from web pages, such as extracting a product offer from a shopping page including product title, description, brand and price. It is an important research topic which has been widely studied in document understanding and web search. Recent natural language models with sequence modeling have demonstrated state-of-the-art performance on web information extraction. However, effectively serializing tokens from unstructured web pages is challenging in practice due to a variety of web layout patterns. Limited work has focused on modeling the web layout for extracting the text fields. In this paper, we introduce WebFormer, a Web-page transFormer model for structure information extraction from web documents. First, we design HTML tokens for each DOM node in the HTML by embedding representations from their neighboring tokens through graph attention. Second, we construct rich attention patterns between HTML tokens and text tokens, which leverages the web layout for effective attention weight computation. We conduct an extensive set of experiments on SWDE and Common Crawl benchmarks. Experimental results demonstrate the superior performance of the proposed approach over several state-of-the-art methods.

CCS CONCEPTS

• **Computing methodologies** → **Information extraction**.

KEYWORDS

web page extraction, structure extraction, transformer

ACM Reference Format:
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 '22), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3485447.3512032

Figure 1: An example of an event web page with its HTML (bottom left) and the extracted structured event information (bottom right), including event title, description, date and time, and location. The corresponding extractions of all text fields are highlighted with colored bounding boxes.

1 INTRODUCTION

Web pages or documents are the most common and powerful source for humans to acquire knowledge. There are billions of websites that contain rich information about various objects. For example, Figure 1 shows a web page describing an event, which contains structured event information including event title, description, date, time and location. The large-scale web data becomes increasingly essential to facilitate new experiences in applications like web search and retrieval, which enables smart assistants to do complex tasks such as "locating kid-friendly events in San Francisco this weekend" and "exploring Nike running shoes less than $50". Therefore, it is an important research problem to extract structured information from web pages.

Structure information extraction from the web [6, 13, 18, 32, 58] is a challenging task due to the unstructured nature of textual data

*Corresponding Authors. This work was done before joining Meta.

WWW '22, April 25–29, 2022, Virtual Event, Lyon, France

ACM ISBN 978-1-4503-9096-5/22/04…$15.00
https://doi.org/10.1145/3485447.3512032

and the diverse layout patterns of the web documents [31, 39]. There has been a lot of interest in this topic, and a plethora of research [10, 40, 55, 59] in this area both in academia and industry. Among the early works, template/wrapper induction [14, 28, 35] has proven to be successful for extracting information from web documents. However, these techniques do not scale to the whole web as obtaining accurate ground truth for all domains is expensive. Moreover, the wrappers go out-of-date quickly because page structure changes frequently, and require periodic updating. One also needs to generate new templates for the new domains.

Recently, learning-based models [17, 46] have been proposed for automatic information extraction. These methods use schema.org markup [41] as the supervision to build machine-learned extractors for different fields. Most recently, with the advance of natural language processing [5, 15, 42], language models with sequence modeling [3, 45] have been applied to web document information extraction. These approaches first sequentialize the web document into a sequence of words, and then use RNN/LSTM [26, 60, 62] or attention networks [21, 52] to extract the text spans corresponding to the structured fields from the sequence. Although existing natural language models achieve promising results on web information extraction, there are several major limitations. First, the structural HTML layout has not been fully exploited, which contains important information and relation about different text fields. For example, in an event page, the event date and location are naturally correlated, which form sibling nodes in the HTML (see Figure 1). In a shopping page, the product price is often mentioned right after the product title on the page. Therefore, encoding the structural HTML beyond sequential modeling is essential in web document extraction. Second, most existing models do not scale up to a large number of fields across domains. They build one separate model for each text field, which are not suitable for large scale extraction, nor can be generalized to new domains. Third, large web documents with long sequences are not modeled effectively. Attention networks, such as Transformer-based models, usually limit their input to 512 tokens due to the quadratic computational cost with the sequence length.

In this paper, we propose WebFormer, a novel Web-page transFormer model that incorporates the HTML layout into the representation of the web document for structure information extraction. WebFormer encodes the field, the HTML and the text sequence in a unified Transformer model. Specifically, we first introduce HTML tokens for each DOM node in the HTML. We then design rich attention patterns for embedding representation among all the tokens. WebFormer leverages the web layout structure for more effective attention weight computation, and therefore explicitly recovers both local syntactic and global layout information of the web document. We evaluate WebFormer on SWDE and Common Crawl benchmarks, which shows superior performance over several state-of-the-art methods. The experimental results also demonstrate the effectiveness of WebFormer in modeling long sequences for large web documents. Moreover, we show that WebFormer is able to extract information on new domains. We summarize the main contributions as follows:

- We propose a novel WebFormer model for structure information extraction from web documents, which effectively integrates the web HTML layout via graph attention.
- We introduce a rich attention mechanism for embedding representation among different types of tokens, which enables the model to encode long sequences efficiently. It also empowers the model for zero-shot extractions on new domains.
- We conduct extensive experiments and demonstrate the effectiveness of the proposed approach over several state-of-the-art baselines.

2 RELATED WORK

2.1 Information Extraction

Early studies of extracting information from the web pages mainly focus on building templates for HTML DOM tree, named wrapper induction [12, 22]. Template extraction techniques have been applied to improve the performance of search engines, clustering, and classification of web pages. They learn desired patterns from the unstructured web documents and construct templates for information extraction. Region extraction methods [7, 39] try to classify portions of a web page according to their specific purposes, e.g., classify whether a text node is the title field. Foley et al. [16] use simple naive-Bayes to classify the web page and SVM methods to get the score for each field. Wang et al. [46] extend this work by designing deep neural network models and using well designed visual features like font sizes, element sizes, and positions.

Recently, there has been an increasing number of works that develop natural language models with sequence modeling [9, 20, 26, 30, 34, 62] for web information extraction. Zheng et al. [60] develop an end-to-end tagging model utilizing BiLSTM, CRF, and attention mechanism without any dictionary. Aggarwal et al. [2] propose a sequence-to-sequence model using an RNN, which leverages relative spatial arrangement of structures. Aghajanyan et al. [3] train a hyper-text language model based on BART [24] on a large-scale web crawl for various downstream tasks. More recently, several attribute extraction approaches [47, 49, 53, 54] have been proposed, which treat each field as an attribute of interest and extract its corresponding value from clean object context such as web title. Chen et al. [9] formulate the web information extraction problem as structural reading comprehension and build a BERT [15] based model to extract structured fields from the web documents. It is worth mentioning that there are also methods that work on multimodal information extraction [44, 45, 48, 56], which focus on extracting the field information from the visual layout or the rendered HTML of the web documents.

2.2 Relation Learning

Relation extraction/learning research [19, 25, 27, 29, 50, 61] is also related to our work. Relation extraction refers to the task of extracting relational tuples and putting them in a knowledge base. Web information extraction can be thought of as the problem where the subject is known (the web document), and given the field (the relation) extract the corresponding text. However, relation extraction has traditionally focused on extracting relations from sentences relying on entity linking systems to identify the subject/object and building models to learn the predicates in a sentence [8, 23]. Whereas in structure information extraction, usually the predicates (the fields) rarely occur in the web documents,

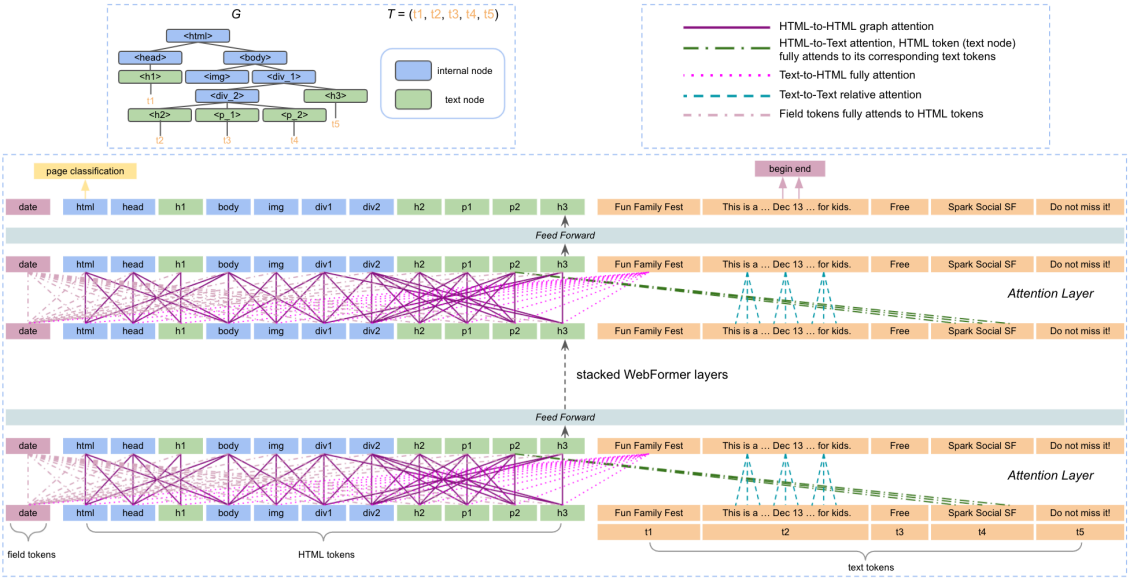

Figure 2: The WebFormer model architecture.

and entity linking is very hard because the domain of all entities is unknown.

3 WEBFORMER

3.1 Problem Definition

We formally define the problem of structure information extraction from web documents in this section. The web document is first processed into a sequence of text nodes and the HTML DOM tree. We denote the text sequence from the web document as $T = (t_1, t_2, \ldots, t_k)$, where t_i represents the i-th text node on the web. k is the total number of text nodes with t_i=$(w_{i_1}, w_{i_2}, \ldots, w_{i_{n_i}})$ as its n_i words/tokens. Note that the ordering of the text nodes does not matter in our model, and one can traverse the DOM tree in any order to obtain all the text nodes. Denote the DOM tree of the HTML as $G = (V, E)$, where V is the set of DOM nodes in the tree with E being the set of edges (see top left in Figure 2). Note that the k text nodes are essentially connected in this DOM representation of the HTML, representing the layout of the web document.

The goal of structure information extraction is that given a set of target fields F=$(f_1, \ldots, f_m)$, extract their corresponding text information from the web document. For example, for the text field "date", we aim to extract the text span "Dec 13" from the web document. Formally, the problem is defined as finding the best text span $\bar{s_j}$ for each field f_j, given the web document T and G:

$$\bar{s_j} = \arg\max_{b_j, e_j} Pr(\, w_{b_j},\ w_{e_j} \mid f_j,\ T,\ G)$$

where b_j and e_j are the begin and end offsets of the extracted text span in the web document for text field f_j.

3.2 Approach Overview

Existing sequence modeling methods either directly model the text sequence from web document [26, 47] or serialize the HTML with the text in a certain order [9, 62] to perform the span based text extraction. In this work, we propose to simultaneously encode the text sequence using the Transformer model and incorporate the HTML layout structure with graph attention.

The overall model architecture is shown in Figure 2. Essentially, our WebFormer model consists of three main components, the input layer, the WebFormer encoder and the output layer. The input layer contains the construction of the input tokens of WebFormer as well as their embeddings, including the field token, the HTML tokens from DOM tree G and the text tokens from the text sequence T. The WebFormer encoder is the main block that encodes the input sequence with rich attention patterns, including HTML-to-HTML (H2H), HTML-to-Text (H2T), Text-to-HTML (T2H) and Text-to-Text (T2T) attentions. In the output layer, the text span corresponding to the field is computed based on the encoded field-dependent text embeddings. We present the detail of each component separately in the following subsections.

3.3 Input Layer

Most previous sequence modeling approaches [2, 53] only encode the text sequence of the web document without utilizing the HTML layout structure. In this work, we jointly model the text sequence with the HTML layout in a unified Transformer model. In particular, we introduce three types of tokens in the input layer of WebFormer.

Field token A set of field tokens are used to represent the text field to be extracted, such as "title", "company" and "base salary" for a job page. By jointly encoding the text field, we are able to construct a unique model across all text fields.

HTML token Each node in the DOM tree G, including both internal nodes (non-text node) and text nodes, corresponds to an HTML token in WebFormer. The embedding of a HTML token can be viewed as a summarization of the sub-tree rooted by this node. For example, in Figure 2, the embedding of the "<*html*>" token

essentially represents the full web document, which can be used for page level classification. On the other hand, the embedding of the text node "<p_2>" summarizes the text sequence t_4.

Text token This is the commonly used word representation in natural language models. For example, t_1 contains three words, "Fun", "Family" and "Fest", which correspond to three text tokens.

In the input layer, every token is converted into a d-dimensional embedding vector. Specifically, for field and text tokens, their final embeddings are achieved by concatenating a word embedding and a segment embedding. For HTML token embedding, they are formulated by concatenating a tag embedding and a segment embedding. The word embedding is widely adopted in the literature [33]. The segment embedding is added to indicate which type the token belongs to, i.e. field, HTML or text. The tag embedding is introduced to represent different HTML-tag of the DOM nodes, e.g. "*div*", "*head*", "*h1*", "*p*", etc. Note that all the embeddings in our approach are trainable. The word embeddings are initialized from the pretrained language model, while the segment and tag embeddings are randomly initialized.

3.4 WebFormer Encoder

The WebFormer encoder is a stack of L identical contextual layers, which efficiently connects the field, HTML and text tokens with rich attention patterns followed by a feed-forward network. The encoder produces effective contextual representations of web documents. To capture the complex HTML layout with the text sequence, we design four different attention patterns, including 1) HTML-to-HTML (H2H) attention which models the relations among HTML tokens via graph attentions. 2) HTML-to-Text (H2T) attention, which bridges the HTML token with its corresponding text tokens. 3) Text-to-HTML (T2H) attention that propagates the information from the HTML tokens to the text tokens. 4) Text-to-Text (T2T) attention with relative position representations. Moreover, WebFormer incorporates the field into the encoding layers to extract the text span for the field.

3.4.1 HTML-to-HTML Attention. The HTML tokens are naturally connected via the DOM tree graph. The H2H attention essentially computes the attention weights among the HTML tokens and transfers the knowledge from one node to another with the graph attention [43]. We use the original graph G that represents the DOM tree structure of the HTML in the H2H attention calculation. In addition, we add edges to connect the sibling nodes in the graph, which is equivalent to include certain neighbors with edge distance 2 in the graph. For example, the HTML token "<*div*1>" is connected with itself, the parent token "<*body*>", the child tokens "<*div*2>" and "<*h*3>", and sibling token "<*img*>". Formally, given the HTML token embedding x_i^H, the H2H graph attention is defined as:

$$\alpha_{ij}^{H2H} = \frac{\exp(e_{ij}^{H2H})}{\sum_{\ell \in \mathcal{N}(x_i^H)} \exp(e_{i\ell}^{H2H})}, \ for\ j \in \mathcal{N}(x_i^H)$$

$$e_{ij}^{H2H} = \frac{x_i^H W_Q^{H2H} (x_j^H W_K^{H2H} + a_{ij}^{H2H})^T}{\sqrt{d}}$$

where $\mathcal{N}(x_i^H)$ indicates the neighbors of the HTML token x_i^H in the graph. W_Q^{H2H} and W_K^{H2H} are learnable weight matrices, and a_{ij}^{H2H} are learnable vectors representing the edge type between the two nodes, i.e. parent, child or sibling. d is the embedding dimension.

3.4.2 HTML-to-Text Attention. The H2T attention is only computed for the text nodes in the HTML to update their contextual embeddings. We adopt a full attention pattern where the HTML token x_i^H is able to attend to each of its text tokens x_j^T in t_i. For example, in Figure 2, the HTML token "<p_2>" attends to all the three text tokens in t_4, i.e. "Spark", "Social" and "SF". The H2T full attention is defined as:

$$\alpha_{ij}^{H2T} = \frac{\exp(e_{ij}^{H2T})}{\sum_{\ell \in t_i} \exp(e_{i\ell}^{H2T})}, \ for\ j \in t_i$$

$$e_{ij}^{H2T} = \frac{x_i^H W_Q^{H2T} (x_j^T W_K^{H2T})^T}{\sqrt{d}}$$

where W_Q^{H2T} and W_K^{H2T} are weight matrices in H2T attention.

3.4.3 Text-to-HTML Attention. In T2H attention, each text token communicates with every HTML token. Intuitively, this T2H attention allows the text token to absorb the high-level representation from these summarization tokens of the web document. The formulation of the T2H attention is analogous to the above H2T attention except that each text token attends to all HTML tokens.

3.4.4 Text-to-Text Attention. The T2T attention is the regular attention mechanism used in various previous models [15, 42], which learns contextual token embeddings for the text sequence. However, the computational cost of the traditional full attention grows quadratically with the sequence length, and thus limits the size of the text tokens. Inspired by the work of [37, 38], our T2T attention adopts relative attention pattern with relative position encodings, where each text token only attends to the text tokens within the same text sequence and within a local radius r. In Figure 2, the local radius r is set to 1, which means each token will only attend to its left and right tokens, and itself. For instance, the text token "is" in t_2 attends to the tokens "This", "is" and "a" within t_2. The formal T2T relative attention is defined as:

$$\alpha_{ij}^{T2T} = \frac{\exp(e_{ij}^{T2T})}{\sum_{i-r \le \ell \le i+r} \exp(e_{i\ell}^{T2T})}, \ for\ i-r \le j \le i+r$$

$$e_{ij}^{T2T} = \frac{x_i^T W_Q^{T2T} (x_j^T W_K^{T2T} + b_{i-j}^{T2T})^T}{\sqrt{d}}$$

where W_Q^{T2T} and W_K^{T2T} are weight matrices in T2T attention. b_{i-j}^{T2T} are learnable relative position encodings representing the relative position between the two text tokens. Note that there are total $2r+1$ possible relative position encodings, i.e. $(i-j) \in \{-r, \ldots, -1, 0, 1, \ldots, r\}$.

3.4.5 Field Token Attention. Our WebFormer model jointly encodes the field information such that the structured fields share the unique encoder. Following the work in [47, 49], we introduce the field tokens into WebFormer and enable full cross-attentions between field and HTML tokens. Note that one can easily add cross-attention between field and text tokens. We found empirically in our experiments that this does not improve the extraction quality. Although there is no direct interaction between field and text tokens,

they are bridged together through the text-to-HTML and the HTML-field attentions.

3.4.6 Overall Attention. We compute the final token representation based on the above rich attention patterns among field, text and HTML tokens. The output embeddings for field, text and HTML tokens z_i^F, z_i^T, z_i^H, are calculated as follows:

$$z_i^F = \sum_j \alpha_{ij}^{F2H} x_j^H W_V^F$$

$$z_i^T = \sum_{i-r \le j \le i+r} \alpha_{ij}^{T2T} x_j^T W_V^T + \sum_k \alpha_{ij}^{T2H} x_k^H W_V^H$$

$$z_i^H = \sum_{j \in \mathcal{N}(x_i^H)} \alpha_{ij}^{H2H} x_j^H W_V^H + \sum_{k \in t_i} \alpha_{ij}^{H2T} x_k^T W_V^T$$

where all the attention weights α_{ij} are described above. W_V^F, W_V^T and W_V^H are the learnable matrices to compute the values for field, text and HTML tokens respectively.

3.5 Output Layer

The output layer of WebFormer extracts the final text span for the field from the text tokens. We apply a softmax function on the output embeddings of the encoder to generate the probabilities for the begin and end indices:

$$P_b = softmax(W_b Z^T),\ P_e = softmax(W_e Z^T)$$

where Z^T is the contextual embedding vectors of the input text sequence. W_b and W_e are two parameter matrices that project the embeddings to the output logits, for the begin and end respectively. Inspired by the work [57], we further predict the end index based on the start index by concatenating the begin token embedding with every token embedding after it.

3.6 Discussion

This section provides discussion that connects WebFormer with previous methods as well as the limitations of our model. If we treat HTML tags as additional text tokens, and combine with the text into a single sequence without the H2H, H2T and T2H attentions, our model architecture degenerates to the sequence modeling approaches [9, 51] that serialize the HTML layout. If we further trim the HTML from the sequence, our model is regressed to the sequence model [47] that only uses the text information. Moreover, if we also remove the text field from the input, our model degenerates to the sequence tagging method [26, 60], which is not able to scale to a large set of target fields.

There are two scenarios where our model is not directly applicable. First, our model focuses on structure information extraction on single object pages, where each target field only has one text value. For a multi-object page, e.g. a multi-event page, there are different titles and dates corresponding to different events on the page, which could be extracted with methods like repeated patterns [1, 46]. Second, there are applications that require to extract information from the rendered pages, where OCR and CNN [51] techniques are used.

Data Splits	SWDE	Common Crawl		
		Events	Products	Movies
Train	99,248	72,367	105,642	57,238
Dev/Test	12,425	9,046	13,205	7,154
Training Time (10 epoch)	4h 15m	3h 46m	4h 22m	3h 21m

Table 1: Statistics of the datasets with the training time.

4 EXPERIMENTS

4.1 Datasets

SWDE [18, 62]: The Structured Web Data Extraction (SWDE) dataset is designed for structural reading comprehension and information extraction on the web. It consists of more than 124,000 web pages from 80 websites of 8 verticals including "auto", "book", "camera", "job", "movie", "nbaplayer", "restaurant" and 'university'. Each vertical consists of 10 websites and contains 3 to 5 target fields of interest. We further split the data into train, dev and test sets with 99,248, 12,425 and 12,425 pages respectively.

Common Crawl[1]: The Common Crawl corpus is widely used in various web search, information extraction and other related tasks. Common Crawl contains more than 250 TiB of content from more than 3 billion web pages. In our experiments, we select web pages that have schema.org annotations[2] within the three domains - **Events**, **Products** and **Movies**. The schema.org annotations contain the website provided markup information about the object, which are used as our ground-truth labels. The fields are {"Name", "Description", "Date", "Location"}, {"Name", "Description", "Brand", "Price", "Color"} and {"Name", "Description", "Genre", "Duration", "Director", "Actor", "Published Date"} for event, product and movie pages respectively. We further filter these pages by restricting to English and single object pages. We downsample the web pages by allowing at most 2,000 pages per website to balance the data, as some websites might dominate, e.g., amazon.com. All datasets are then randomly split into train, dev and test sets with raito 8:1:1. The details are given in Table 1.

4.2 Implementation Detail

For data pre-processing, we use open-source LXML library[3] to process each page for obtaining the DOM tree structures. We then use in order traverse of the DOM tree to obtain the text nodes sequence. We implemented our models using Tensorflow and Keras. Each model is trained on a 32 core TPU v3 configuration. The word embedding is initialized with the pretrained BERT-base. The parameters used in WebFormer are 12 layers, 768 hidden size, 3072 hidden units (for FFN) and 64 local radius. The maximum text sequence length is set to 2048. The maximum number of HTML tokens are set to 256. During training, we use the gradient descent algorithm with Adam optimizer. The initial learning rate is set to $3e^{-5}$. The batch size for each update is set as 64 and the model is trained for up to 10 epochs. The dropout probability for the attention layer is set to 0.1.

[1] http://commoncrawl.org/connect/blog/
[2] https://schema.org/
[3] https://lxml.de/

Models	SWDE		Common Crawl					
			Events		Products		Movies	
	EM	F1	EM	F1	EM	F1	EM	F1
OpenTag	81.33 ± 0.22	86.54 ± 0.27	77.14 ± 0.26	83.71 ± 0.12	72.57 ± 0.20	77.75 ± 0.19	80.36 ± 0.15	85.06 ± 0.18
DNN	80.53 ± 0.15	85.64 ± 0.26	78.43 ± 0.18	85.06 ± 0.21	74.64 ± 0.27	78.56 ± 0.15	82.44 ± 0.23	86.65 ± 0.16
AVEQA	83.27 ± 0.32	88.75 ± 0.16	80.82 ± 0.21	86.47 ± 0.14	74.85 ± 0.32	79.49 ± 0.28	83.87 ± 0.30	88.51 ± 0.19
SimpDOM	84.67 ± 0.23	90.35 ± 0.21	81.96 ± 0.24	86.33 ± 0.17	75.12 ± 0.27	78.22 ± 0.21	82.59 ± 0.25	87.72 ± 0.18
H-PLM	83.42 ± 0.20	89.04 ± 0.18	82.65 ± 0.15	87.52 ± 0.17	76.24 ± 0.17	81.13 ± 0.26	83.72 ± 0.26	89.34 ± 0.17
WebFormer	**86.58 ± 0.16**	**92.46 ± 0.24**	**84.79 ± 0.24**	**89.33 ± 0.18**	**80.67 ± 0.20**	**83.37 ± 0.23**	**85.30 ± 0.19**	**90.41 ± 0.24**

Table 2: Performance comparison on all datasets. Results are statistically significant with p-value < 0.001.

Fields	Events		Products		Movies	
	EM	F1	EM	F1	EM	F1
Name	88.27	93.46	85.11	90.53	89.32	93.57
Description	81.62	85.50	77.94	81.46	82.71	88.19
Date	86.86	91.48	-	-	-	-
Location	82.41	86.88	-	-	-	-
Brand	-	-	84.23	85.63	-	-
Price	-	-	75.65	76.86	-	-
Color	-	-	80.42	82.35	-	-
Genre	-	-	-	-	89.49	92.67
Duration	-	-	-	-	83.74	88.35
Director	-	-	-	-	86.28	91.38
Actor	-	-	-	-	80.16	87.44
Publish Date	-	-	-	-	85.40	91.27

Table 3: Field level metrics of WebFormer.

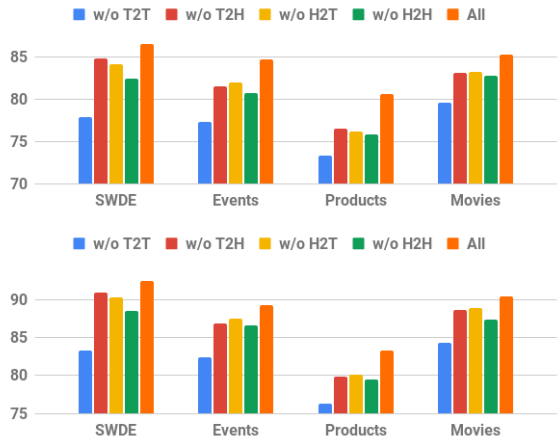

Figure 3: Results of WebFormer with different attention patterns. Top: EM scores. Bottom: F1 scores.

4.3 Evaluation Metric

We evaluate the performance of the WebFormer model with two standard evaluation metrics: **Exact Match** (EM) and **F1** from the package released in [36]. Exact Match is used to evaluate whether a predicted span is completely the same as the ground truth. It will be challenging for those answers that are only part of the text. F1 measures the overlap of the extracted answer and the ground truth by splitting the answer span into tokens and compute F1 score on them. We repeat each experiment 10 times and report the metrics on the test sets based on the average over these runs.

4.4 Baselines

OpenTag [60] uses a BiLSTM-Attention-CRF architecture with sequence tagging strategies. OpenTag does not encode the field and thus builds one model per field.

DNN [46] applies deep neural networks for information extraction. Text nodes in the HTML are treated as candidates, and are extracted with DNN classifiers.

AVEQA [47] formulates the problem as an attribute value extraction task, where each field is treated as an attribute. This model jointly encodes both the attribute and the document with a BERT [15] encoder.

SimpDOM [62] treats the problem as DOM tree node tagging task by extracting the features for each text node including XPath, and uses a LSTM to jointly encode with the text features.

H-PLM [9] sequentializes the HTML together with the text and builds a sequence model using the pre-training ELECTRA [11] as backbone.

The codes for OpenTag[4] and H-PLM[5] are publicly available. For our previous works DNN and AVEQA, we use the original codes for the papers. For SimpDOM, we re-implement their model using the parameters from the paper.

4.5 Results and Discussion

4.5.1 Performance Comparison. The evaluation results of WebFormer and all baselines are reported in Table 2. From these comparison results, we can see that WebFormer achieves the best performance among all compared methods on all datasets. For example, the EM metric of WebFormer increases over 7.8% and 5.8% compared with AVEQA and H-PLM on Products. There are three main reasons: First, our model integrates the HTML layout into a unified HTML-text encoder with rich attention, which enables the model to effectively understand the web layout structure. Second, WebFormer adopts the relative position encoding in T2T attention, which allows our model to represent large documents efficiently. Third, the field information is jointly encoded and attended with both HTML and text tokens. Different fields share one encoder and thus are able to benefit from each other. We further report the field level results of WebFormer on the Common Crawl dataset in Table 3. It can be seen that some fields, such as "Name" and "Genre", obtain

[4]https://github.com/hackerxiaobai/OpenTag_2019
[5]https://github.com/X-LANCE/WebSRC-Baseline

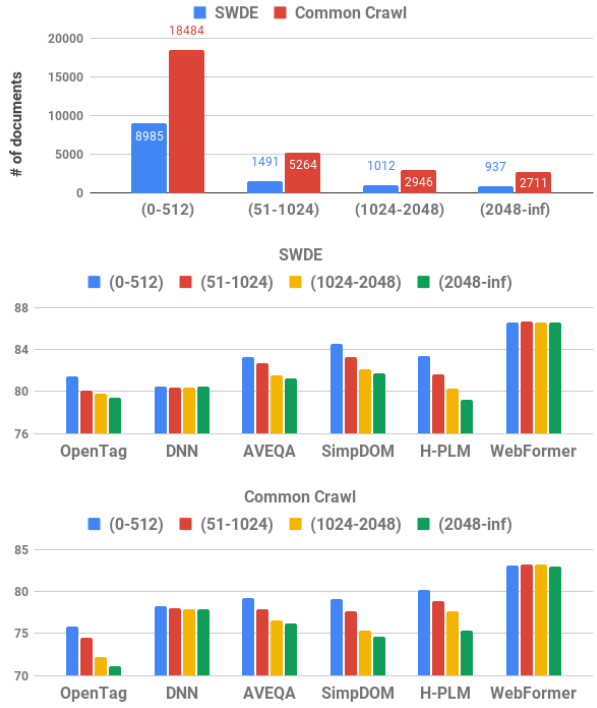

Figure 4: EM scores of different methods within each bucket of sequence length.

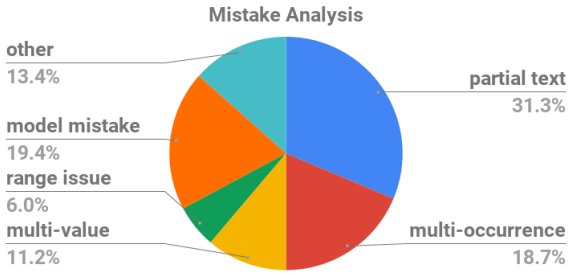

Figure 5: Mistake analysis: distribution of different type of mistakes.

	parameters	SWDE	Common Crawl
AVEQA	110M	83.27	78.45
H-PLM	110M	83.42	80.78
WebFormer-2L	45M	82.05	76.73
WebFormer-6L	82M	83.86	79.35
WebFormer-12L-share	109M	85.29	81.49
WebFormer-12L	151M	86.58	83.22
WebFormer-24L	285M	**87.84**	**86.51**

Table 4: EM results over different model configurations.

relatively higher scores compared with "Price" and "Location". We also observe that the difference between EM and F1 scores is very small for fields like "Brand" and "Color". The reason is that their text spans are usually very short, containing just one or two tokens.

4.5.2 Impact of Rich Attentions. To understand the impact of the rich attention patterns, we conduct a set of experiments by removing each attention from our model. Specifically, we train four separate models without T2T, H2T, T2H and H2H attention respectively. The results of these four models and WebFormer (refer to All) on all datasets are shown in Figure 3. It is not surprising to see that the performance drops significantly without the T2T local attention. The reason is that T2T is used to model the contextual token embeddings for the text sequence, which is the fundamental component in the Transformer model. We can also observe that the model without H2H graph attention achieves much worse performance compared to the models without T2H or H2T attention. This observation validates that the HTML layout information encoded within the H2H attention is crucial for extracting structure fields from web documents. Moreover, it is clear that WebFormer with T2H and H2T attentions further improve the model performance on all datasets.

4.5.3 Impact on Large Document. To evaluate the impact of different models on large documents with long text sequence, we group the test examples into four buckets w.r.t. the sequence length of the example (i.e. 0-512, 512-1024, 1024-2048 and 2048-inf), and compute the metrics in each bucket for all methods. The length distribution of the test documents with EM scores on both datasets (for Common Crawl, we merge all the test sets from Events, Products and Movies) are shown in Figure 4. It can be seen that WebFormer achieves consistent results w.r.t. the sequence length. In contrast, the performances of OpenTag, AVEQA, SimpDOM and H-PLM go down with the increasing of the sequence length. Our hypothesis is that WebFormer utilizes L2L relative attention and the H2L attention, which enables the model to encode web documents with long sequences effectively and efficiently. Note that the DNN model does not depend on the sequence length and thus does not suffer from the long sequence.

4.5.4 Error Analysis. We conduct error analysis of WebFormer over 160 and 60 randomly selected Exact Match mistakes on SWDE and Common Crawl dataset respectively (5 per field). We identify several major mistake patterns and summarize them here: 1) Partial text extraction: The largest group of mistakes is that our model extracts a substring of the ground-truth text. For example, our model extracts "Fun Festival" as the event name instead of "Fun Festival at Square Park". 2) Multiple occurrences issue: There are cases where the target field is mentioned multiple times on the web page. For example, our model extracts "SEP 11" as the date, but the ground-truth text is "Sat, September 11, 2011". 3) Multi-value issue: The other type of error is that the field has multiple values and we only extract one of them. For example, a product has both "blue" and "white" as its color where we only extract "blue". 4) Range issue: There are a certain amount of mistakes that fall into the range issue group. For instance, our model extracts the "price" as "19.90" from the ground-truth "19.90 - 26.35" which is a range of prices. 5) Model mistakes: There are few other extraction errors made by the model,

batch size	64			128			512		
learning rate	$3x10^{-5}$	$5x10^{-5}$	$1x10^{-4}$	$3x10^{-5}$	$5x10^{-5}$	$1x10^{-4}$	$3x10^{-5}$	$5x10^{-5}$	$1x10^{-4}$
SWDE	**86.58**	86.36	86.20	86.37	86.42	86.35	86.18	86.11	86.28
Events	**84.79**	84.62	NaN	84.54	84.46	84.65	84.11	84.27	84.13
Products	80.67	**80.71**	NaN	80.32	80.38	80.40	79.96	80.23	80.37
Movies	**85.30**	85.21	85.14	84.58	84.75	84.83	84.39	84.56	84.77

Table 5: EM results of WebFormer with different batch sizes and learning rates on all datasets.

which are hard cases even for human raters. The summarization of the mistake analysis is reported in Figure 5. By looking closely at these mistake patterns, we observe that our model actually extracts the correct or partially correct answers for most cases in the group of 1), 2), 3) and 4). These mistakes can be easily fixed by marking all answer occurrences and values as positives in the training, and adopting a BIO-based span extraction as mentioned. However, there are still difficult cases which require further investigations into the training data and the model.

4.5.5 Ablation Study. We further conduct a series of ablation studies of WebFormer. The WebFormer base model contains 12 layers. We first evaluate our model with a different number of encoder layers, i.e. 2L, 6L and 24L. We also evaluate another ablation of WebFormer by sharing the model parameters. Specifically, the query matrices of the text and HTML tokens are shared, i.e. $W_Q^{T2T}=W_Q^{T2H}=W_Q^T$, $W_Q^{H2H}=W_Q^{H2T}=W_Q^H$, $W_K^{T2T}=W_K^{H2T}=W_K^T$ and $W_K^{H2H}=W_K^{T2H}=W_K^H$. This model is referred to as WebFormer-12L-share. The EM results with the number of model parameters are shown in Table 4. It can be observed that WebFormer-24L achieves the best performance, which is consistent with our expectations. Similar behavior is also observed in [4, 15]. However, a larger model usually requires longer training time, as well as inference. The training time of the base models are reported in Table 1.

4.5.6 Impact of Training Batch Size and Learning Rate. To evaluate the model performance with different training batch size and learning rate, we conduct experiments to train a set of WebFormer models with a hyper-parameter sweep consisting of learning rates in {$3x10^{-5}$, $5x10^{-5}$, $1x10^{-4}$} and batch-size in {64, 128, 512} on the training set. The EM results with different learning rates and batch sizes on all datasets are reported in Table 5. It can be seen from the tables that WebFormer achieves the best result with batch size 64 and learning rate $3x10^{-5}$ on all datasets except Products. The observation is consistent with the findings in work [47], where smaller batch size usually leads to better performance. This is also the reason that we set batch size to 64 and learning rate to $3x10^{-5}$ in all our previous experiments.

4.5.7 Zero-shot/Few-shot Extraction. We conduct zero-shot and few-shot extraction experiments to evaluate the generalization ability of WebFormer on unseen domains/fields. In this experiment, we first pre-train a WebFormer model on Products and Movies data only. We then perform fine-tuning on Events data for 10K steps by varying the number of training examples from {0, 1, 2, 5, 10, 50, 100}. The EM scores of WebFormer on all four event fields are shown in Figure 6. There are several interesting observations from

Figure 6: EM results on zero-shot and few-shot learning.

this table. First, when the number of training examples is 0 (zero-shot extraction), the EM scores on "Name" and "Description" are reasonable around 75%. However, the score on "Location" is close to 0. The reason is that both "Name" and "Description" are general fields that appear across domains, e.g. they both present in Products and Movies data. Therefore, the learned knowledge in WebFormer can be directly transferred to a new domain - Events. On the other hand, the pretrained model lacks knowledge about "Location" and thus performs poorly on this field. Second, it is not surprising to see that the EM scores increase with more training examples, and reach reasonably high values with 100 training examples. We also observe that the EM score for "Location" boosts dramatically even with one or two training examples.

5 CONCLUSION

In this paper, we introduce a novel Web-page transFormer model, namely WebFormer, for structure information extraction from web documents. The structured HTML layout information is jointly encoded through the rich attention patterns with the text information. WebFormer effectively recovers both local syntactic and global layout information from web document serialization. An extensive set of experimental results on SWDE and Common Crawl benchmarks has demonstrated the superior performance of the proposed approach over several state-of-the-art methods. In future, we plan to extend this work to multimodal learning that incorporates visual features.

ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China (No. 62176270).

REFERENCES

[1] Marco D. Adelfio and Hanan Samet. 2013. Schema Extraction for Tabular Data on the Web. *PVLDB* 6, 6 (2013), 421–432.

[2] Milan Aggarwal, Hiresh Gupta, Mausoom Sarkar, and Balaji Krishnamurthy. 2020. Form2Seq : A Framework for Higher-Order Form Structure Extraction. In *EMNLP*. 3830–3840.

[3] Armen Aghajanyan, Dmytro Okhonko, Mike Lewis, Mandar Joshi, Hu Xu, Gargi Ghosh, and Luke Zettlemoyer. 2021. HTLM: Hyper-Text Pre-Training and Prompting of Language Models. *CoRR* abs/2107.06955 (2021).

[4] Joshua Ainslie, Santiago Ontañón, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. 2020. ETC: Encoding Long and Structured Inputs in Transformers. In *EMNLP*. 268–284.

[5] Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The Long-Document Transformer. *CoRR* abs/2004.05150 (2020).

[6] Andrew Carlson and Charles Schafer. 2008. Bootstrapping Information Extraction from Semi-structured Web Pages. In *ECML/PKDD*. 195–210.

[7] Chia-Hui Chang, Mohammed Kayed, Moheb R. Girgis, and Khaled F. Shaalan. 2006. A Survey of Web Information Extraction Systems. *IEEE Trans. Knowl. Data Eng.* 18, 10 (2006), 1411–1428.

[8] Ke Chen, Lei Feng, Qingkuang Chen, Gang Chen, and Lidan Shou. 2019. EXACT: Attributed Entity Extraction By Annotating Texts. In *SIGIR*. 1349–1352.

[9] Xingyu Chen, Zihan Zhao, Lu Chen, Jiabao Ji, Danyang Zhang, Ao Luo, Yuxuan Xiong, and Kai Yu. 2021. WebSRC: A Dataset for Web-Based Structural Reading Comprehension. In *EMNLP*. 4173–4185.

[10] Mengli Cheng, Minghui Qiu, Xing Shi, Jun Huang, and Wei Lin. 2020. One-shot Text Field labeling using Attention and Belief Propagation for Structure Information Extraction. In *ACM MM*. 340–348.

[11] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. In *ICLR*.

[12] William W. Cohen, Matthew Hurst, and Lee S. Jensen. 2002. A flexible learning system for wrapping tables and lists in HTML documents. In *WWW*. 232–241.

[13] Valter Crescenzi and Giansalvatore Mecca. 2004. Automatic information extraction from large websites. *J. ACM* 51, 5 (2004), 731–779.

[14] Nilesh N. Dalvi, Ravi Kumar, and Mohamed A. Soliman. 2011. Automatic Wrappers for Large Scale Web Extraction. *Proc. VLDB Endow.* 4, 4 (2011), 219–230.

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*. 4171–4186.

[16] John Foley, Michael Bendersky, and Vanja Josifovski. 2015. Learning to Extract Local Events from the Web. In *SIGIR*. 423–432.

[17] Tomas Gogar, Ondrej Hubácek, and Jan Sedivý. 2016. Deep Neural Networks for Web Page Information Extraction. In *AIAI*, Vol. 475. 154–163.

[18] 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 *SIGIR*. 775–784.

[19] Zhengqiu He, Wenliang Chen, Zhenghua Li, Meishan Zhang, Wei Zhang, and Min Zhang. 2018. SEE: Syntax-Aware Entity Embedding for Neural Relation Extraction. In *AAAI*. 5795–5802.

[20] Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF Models for Sequence Tagging. *CoRR* abs/1508.01991 (2015).

[21] Wonseok Hwang, Jinyeong Yim, Seunghyun Park, Sohee Yang, and Minjoon Seo. 2021. Spatial Dependency Parsing for Semi-Structured Document Information Extraction. In *ACL/IJCNLP*. 330–343.

[22] Chulyun Kim and Kyuseok Shim. 2011. TEXT: Automatic Template Extraction from Heterogeneous Web Pages. *IEEE Trans. Knowl. Data Eng.* 23, 4 (2011), 612–626.

[23] Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-Shot Relation Extraction via Reading Comprehension. In *CoNLL*. 333–342.

[24] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *ACL*. 7871–7880.

[25] Yang Li, Guodong Long, Tao Shen, Tianyi Zhou, Lina Yao, Huan Huo, and Jing Jiang. 2020. Self-Attention Enhanced Selective Gate with Entity-Aware Embedding for Distantly Supervised Relation Extraction. In *AAAI*. 8269–8276.

[26] Bill Yuchen Lin, Ying Sheng, Nguyen Vo, and Sandeep Tata. 2020. FreeDOM: A Transferable Neural Architecture for Structured Information Extraction on Web Documents. In *SIGKDD*. 1092–1102.

[27] Jie Liu, Shaowei Chen, Bingquan Wang, Jiaxin Zhang, Na Li, and Tong Xu. 2020. Attention as Relation: Learning Supervised Multi-head Self-Attention for Relation Extraction. In *IJCAI*. 3787–3793.

[28] Colin Lockard, Xin Luna Dong, Prashant Shiralkar, and Arash Einolghozati. 2018. CERES: Distantly Supervised Relation Extraction from the Semi-Structured Web. *Proc. VLDB Endow.* 11, 10 (2018), 1084–1096.

[29] Colin Lockard, Prashant Shiralkar, Xin Luna Dong, and Hannaneh Hajishirzi. 2020. ZeroShotCeres: Zero-Shot Relation Extraction from Semi-Structured Webpages. In *ACL*. 8105–8117.

[30] Xuezhe Ma and Eduard H. Hovy. 2016. End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. In *ACL*. 1064–1074.

[31] Bodhisattwa Prasad Majumder, Navneet Potti, Sandeep Tata, James Bradley Wendt, Qi Zhao, and Marc Najork. 2020. Representation Learning for Information Extraction from Form-like Documents. In *ACL*. 6495–6504.

[32] Tomohiro Manabe and Keishi Tajima. 2015. Extracting Logical Hierarchical Structure of HTML Documents Based on Headings. *Proc. VLDB Endow.* 8, 12 (2015), 1606–1617.

[33] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS*. 3111–3119.

[34] Rafal Powalski, Lukasz Borchmann, Dawid Jurkiewicz, Tomasz Dwojak, Michal Pietruszka, and Gabriela Palka. 2021. Going Full-TILT Boogie on Document Understanding with Text-Image-Layout Transformer. *CoRR* abs/2102.09550 (2021).

[35] Julia Proskurnia, Marc-Allen Cartright, Lluis Garcia Pueyo, Ivo Krka, James B. Wendt, Tobias Kaufmann, and Balint Miklos. 2017. Template Induction over Unstructured Email Corpora. In *WWW*. 1521–1530.

[36] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100, 000+ Questions for Machine Comprehension of Text. In *EMNLP*. 2383–2392.

[37] Peter Shaw, Philip Massey, Angelica Chen, Francesco Piccinno, and Yasemin Altun. 2019. Generating Logical Forms from Graph Representations of Text and Entities. In *ACL*. 95–106.

[38] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-Attention with Relative Position Representations. In *NAACL-HLT*. 464–468.

[39] Hassan A. Sleiman and Rafael Corchuelo. 2013. A Survey on Region Extractors from Web Documents. *IEEE Trans. Knowl. Data Eng.* 25, 9 (2013), 1960–1981.

[40] Guozhi Tang, Lele Xie, Lianwen Jin, Jiapeng Wang, Jingdong Chen, Zhen Xu, Qianying Wang, Yaqiang Wu, and Hui Li. 2021. MatchVIE: Exploiting Match Relevancy between Entities for Visual Information Extraction. In *IJCAI*. 1039–1045.

[41] Nicolas Tempelmeier, Elena Demidova, and Stefan Dietze. 2018. Inferring Missing Categorical Information in Noisy and Sparse Web Markup. In *WWW*. 1297–1306.

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NIPS*. 5998–6008.

[43] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.

[44] Daheng Wang, Prashant Shiralkar, Colin Lockard, Binxuan Huang, Xin Luna Dong, and Meng Jiang. 2021. TCN: Table Convolutional Network for Web Table Interpretation. In *WWW*. 4020–4032.

[45] Jiapeng Wang, Tianwei Wang, Guozhi Tang, Lianwen Jin, Weihong Ma, Kai Ding, and Yichao Huang. 2021. Tag, Copy or Predict: A Unified Weakly-Supervised Learning Framework for Visual Information Extraction using Sequences. In *IJCAI*. 1082–1090.

[46] Qifan Wang, Bhargav Kanagal, Vijay Garg, and D. Sivakumar. 2019. Constructing a Comprehensive Events Database from the Web. In *CIKM*. 229–238.

[47] Qifan Wang, Li Yang, Bhargav Kanagal, Sumit Sanghai, D. Sivakumar, Bin Shu, Zac Yu, and Jon Elsas. 2020. Learning to Extract Attribute Value from Product via Question Answering: A Multi-task Approach. In *SIGKDD*. 47–55.

[48] Benfeng Xu, Quan Wang, Yajuan Lyu, Yong Zhu, and Zhendong Mao. 2021. Entity Structure Within and Throughout: Modeling Mention Dependencies for Document-Level Relation Extraction. In *AAAI*. 14149–14157.

[49] Huimin Xu, Wenting Wang, Xin Mao, Xinyu Jiang, and Man Lan. 2019. Scaling up Open Tagging from Tens to Thousands: Comprehension Empowered Attribute Value Extraction from Product Title. In *ACL*. 5214–5223.

[50] Wang Xu, Kehai Chen, and Tiejun Zhao. 2021. Document-Level Relation Extraction with Reconstruction. In *AAAI*. 14167–14175.

[51] Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. 2020. LayoutLM: Pre-training of Text and Layout for Document Image Understanding. In *SIGKDD*. 1192–1200.

[52] Yang Xu, Yiheng Xu, Tengchao Lv, Lei Cui, Furu Wei, Guoxin Wang, Yijuan Lu, Dinei A. F. Florêncio, Cha Zhang, Wanxiang Che, Min Zhang, and Lidong Zhou. 2021. LayoutLMv2: Multi-modal Pre-training for Visually-rich Document Understanding. In *ACL/IJCNLP*. 2579–2591.

[53] Jun Yan, Nasser Zalmout, Yan Liang, Christan Grant, Xiang Ren, and Xin Luna Dong. 2021. AdaTag: Multi-Attribute Value Extraction from Product Profiles with Adaptive Decoding. In *ACL/IJCNLP*. 4694–4705.

[54] Li Yang, Qifan Wang, Zac Yu, Anand Kulkarni, Sumit Sanghai, Bin Shu, Jon Elsas, and Bhargav Kanagal. 2022. MAVE: A Product Dataset for Multi-Source Attribute Value Extraction. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. 1256–1265.

[55] Liu Yang, Mingyang Zhang, Cheng Li, Michael Bendersky, and Marc Najork. 2020. Beyond 512 Tokens: Siamese Multi-depth Transformer-based Hierarchical Encoder for Long-Form Document Matching. In *CIKM*. 1725–1734.

[56] Xiao Yang, Ersin Yumer, Paul Asente, Mike Kraley, Daniel Kifer, and C. Lee Giles. 2017. Learning to Extract Semantic Structure from Documents Using Multimodal

Fully Convolutional Neural Networks. In *CVPR*. 4342–4351.
[57] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *NeurIPS*. 5754–5764.
[58] Junlang Zhan and Hai Zhao. 2020. Span Model for Open Information Extraction on Accurate Corpus. In *AAAI*. 9523–9530.
[59] Kai Zhang, Yuan Yao, Ruobing Xie, Xu Han, Zhiyuan Liu, Fen Lin, Leyu Lin, and Maosong Sun. 2021. Open Hierarchical Relation Extraction. In *NAACL-HLT*. 5682–5693.
[60] Guineng Zheng, Subhabrata Mukherjee, Xin Luna Dong, and Feifei Li. 2018. OpenTag: Open Attribute Value Extraction from Product Profiles. In *SIGKDD*. 1049–1058.
[61] Hao Zheng, Zhoujun Li, Senzhang Wang, Zhao Yan, and Jianshe Zhou. 2016. Aggregating Inter-Sentence Information to Enhance Relation Extraction. In *AAAI*. 3108–3115.
[62] Yichao Zhou, Ying Sheng, Nguyen Vo, Nick Edmonds, and Sandeep Tata. 2021. Simplified DOM Trees for Transferable Attribute Extraction from the Web. *CoRR* abs/2101.02415 (2021).