Simplified DOM Trees for Transferable Attribute Extraction from Web Documents

Anonymous authors

ABSTRACT

There has been a steady need to precisely extract structured knowledge from the web (i.e. HTML documents). Given a web page, extracting a structured object along with various attributes of interest (e.g. price, publisher, author, and genre for a book) can facilitate a variety of downstream applications such as large-scale knowledge base construction, e-commerce product search, and personalized recommendation. Considering each web page is rendered from an HTML DOM tree, existing approaches formulate the problem as a DOM tree node tagging task. However, they either rely on computationally expensive visual feature engineering or are incapable of modeling the relationship among the tree nodes. In this paper, we propose a novel transferable method, Simplified DOM Trees for Attribute Extraction (SimpDOM), to tackle the problem by efficiently retrieving useful context for each node by leveraging the tree structure. We study two challenging experimental settings: (i) intra-vertical few-shot extraction, and (ii) cross-vertical few-shot extraction with out-of-domain knowledge, to evaluate our approach. Extensive experiments on the SWDE public dataset show that SimpDOM outperforms the state-of-the-art (SOTA) method by 1.44% on the F1 score. We also find that utilizing knowledge from a different vertical (cross-vertical extraction) is surprisingly useful and helps beat the SOTA by a further 1.37%.

CCS CONCEPTS

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

KEYWORDS

structured data extraction, web information extraction

ACM Reference Format:

and Anonymous authors. 2021. Simplified DOM Trees for Transferable Attribute Extraction from Web Documents. In *Proceedings of The Web Conference 2021 (WWW '21), April 19-23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

As the world wide web explosively grows nowadays, there has been a perennial need to automate the translation of web pages into structured knowledge [6, 15]. Attribute extraction systems recognize attributes of interest from web pages. For example, collecting book authors can facilitate the user's faceted search by allowing users to narrow down the search results with a filter on the book's attribute. Attribute extraction as well enables various downstream applications including large-scale knowledge base/graph construction [12, 40], e-commerce product search [4, 15], and personalized recommendation [39]. However, the semi-structured data format, noisy page contents, and multifarious page layouts all make it a non-trivial task to extract attributes, compared to the unstructured texts, which can be easily modeled as a sequence [24].

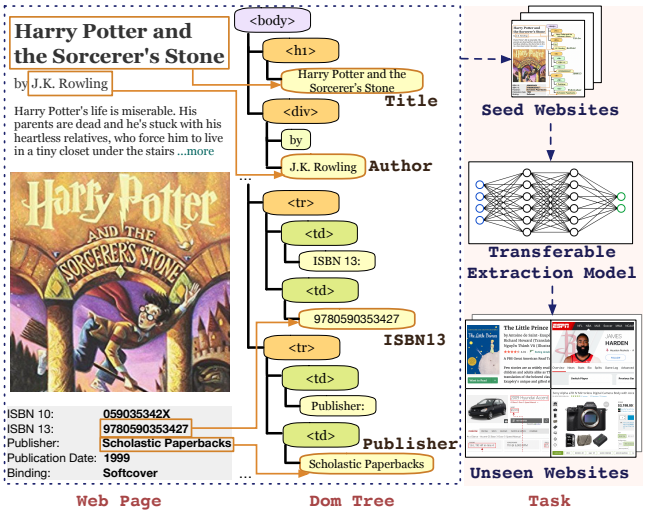

Figure 1: Learning a transferable model based on HTML DOM trees to extract attributes from unseen websites of various verticals. Note that every web page is rendered from a Document Object Model (DOM) Tree. All the attributes are located in the leaf nodes of the trees.

Take Figure 1 as an example. The web page on the left-hand-side is a partial screenshot from a bookstore website. The web page is rendered to display in a browser based on the source data, a Document Object Model (DOM) tree [14] (a corresponding subtree is shown in the middle of Figure 1). In this paper, our goal is to extract attributes of interest such as {*title, author, isbn13, publisher*} from the detail pages of various websites. A detail page denotes a page that corresponds to a single data record [5], like a book in a bookstore or an NBA player on a sports website.

Traditional solutions rely on the fact that many websites are created by templates such as Wrapper Induction [2, 19, 27]. Some unsupervised methods [7, 43] avoid the use of templates and can automatically extract attributes, but neglect the semantics of attribute values. Thus considerable human efforts are required for either periodically updating templates or annotating unseen websites. In this

WWW '21, April 19-23, 2021 , Ljubljana, Slovenia

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

work, we aim to build a novel transferable model to reduce expensive human efforts and to extract attributes from unseen websites of various verticals.

Some recent work [15, 25] explores visual patterns of each node such as its bounding box coordinates and surrounding nodes on the web page. However, achieving these features requires a computationally expensive rendering process and extra memory space to save the necessary images, CSS, and JavaScript files that could easily be out-of-date. FreeDOM [21] avoids rendering-based features and models the pairwise node relationship with node-level feature representations that are learned separately. Nevertheless, it is inconvenient to deploy such a two-stage model in practice. The rich DOM tree-level contexts are neglected by this method as well. In this paper, we propose a novel single-stage approach, Simplified DOM Trees for Attribute Extraction (SimpDOM), that does not require visual features, instead relying on a careful construction of the context of a node in the DOM tree that generalizes well to unseen websites in the domain[1] as well as to websites in other domains.

Specifically, SimpDOM builds a rich representation for each node by focusing on its contextual features. Then a node classification is conducted to decide which attribute type it belongs to. For instance in Figure 1, we notice that the closest text node to "J. K. Rowling" contains information "by" which means "J. K. Rowling" is likely to be the *author* of this book. We also notice important attribute values are usually clustered together to draw readers' attention, like *isbn13*, *publisher*, and *publication date* appear right in the same table. In short, the contexts in a simplified neighborhood can provide website-invariant features such as some semantically informative expressions and vertical-invariant clues such as the co-occurrence of multiple attribute values. We visualize the neighboring relationship of DOM nodes for three websites from two verticals in figure 2. Obviously, the nodes that contain attribute values are always close to each other in the DOM trees.

We consider two challenging experimental scenarios in this paper, (i) intra-vertical few-shot extraction, where we learn a model with a few labeled seed websites and predict on other unseen websites from the same vertical; (ii) cross-vertical few-shot extraction with out-of-domain knowledge, where we train the model with all the websites from an out-of-domain vertical *A*, then finetune this model with a few seed sites from vertical *B*, and finally test it on other unseen websites from vertical *B*. The first scenario tests the transferability among websites from the same vertical while the second assesses the effectiveness of cross-domain knowledge.

Overall, our paper describes the following contributions:

- To the best of our knowledge, this is the first work that efficiently extracts each node's informative contexts from the DOM trees to tackle the attribute extraction task.
- We are also the first to study the transferable representations for the cross-vertical few-shot attribute extraction scenario.
- Extensive experiments on the public dataset, SWDE [15], show that SimpDOM significantly outperforms the SOTA method by 1.44% on the F1 score, and the out-of-domain knowledge helps beat the SOTA by a further 1.37%.
- We will open-source our implementations to provide a testbed and facilitate future research in this direction.

[1]We use domain and vertical interchangeable in this paper.

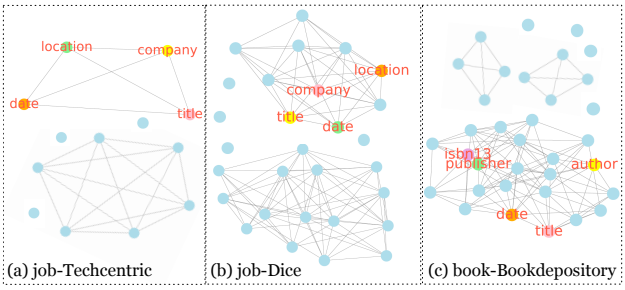

Figure 2: Graph visualization of the DOM node neighborhood. Each node is linked to its close neighbors depending on the DOM tree structures. A site-invariant feature can be concluded that nodes containing attribute values are usually close to each other in the DOM trees.

2 PROBLEM FORMULATION AND APPROACH

In this section, we formally define the problem and introduce the outline of our proposed method, SimpDOM.

2.1 Few-shot attribute extraction from semi-structured websites

We tackle the problem of extracting attributes from unseen semi-structured websites. Each *vertical* V has a set of websites. Each *website* W is composed of a collection of *detailed pages* which share a similar template. Each page has a DOM tree T which contains a variable node set X and a fixed node set Y where the text contents are stored, and also a set of non-text nodes Z. Fixed nodes remain the same across different detailed pages on the same website while variable nodes may contain different contents.

Attribute Extraction. The goal of attribute extraction is to extract a possible value for each attribute type from the DOM tree nodes. We narrow down the search range to variable nodes because the attribute values should vary in different detailed pages. We formulate the attribute extraction as a node tagging task. Given a detailed page p with a set of variable nodes X, we aim to learn a model to classify each node $x \in X$ into one of the pre-defined vertical-specific types (e.g. *title, author, isbn13, publisher*) or *none* representing that this node does not contain any attribute values. We assume that one node can correspond to at most one pre-defined attribute type [15].

Few-shot Intra-vertical Extraction. Given a set of annotated seed websites $\{W_1^a, W_2^a, ..., W_i^a\}$ from vertical V, we aim to learn a transferable model $\mathcal{M}$ to extract attributes from a larger set of unseen websites $\{W_1^u, W_2^u, ..., W_j^u\}$ from the same vertical.

Few-shot Cross-vertical Extraction. In this scenario, we leverage a set of annotated out-of-domain websites from vertical V_1 to learn a transferable extraction model $\mathcal{M}_o$ and fine-tune the model with seed websites $\{W_1^a, W_2^a, ..., W_i^a\}$ from vertical V_2. Finally, we extract attributes from unseen websites $\{W_1^u, W_2^u, ..., W_j^u\}$ of V_2.

2.2 Approach Overview

Figure 3 shows the overall framework of the proposed SimpDOM model for the few-shot attribute extraction task. We firstly simplify

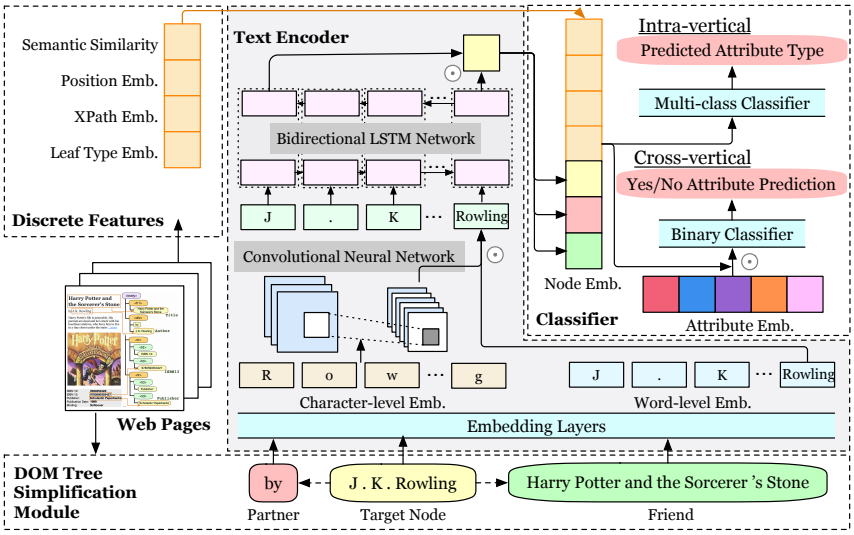

Figure 3: The overall architecture of SimpDOM. Nodes' textual features are encoded by LSTM and CNN at the word-level and character-level respectively. A set of discrete features are built from the DOM trees including leaf type, XPath, and the relative position of each node. $[\cdot \odot \cdot]$ denotes vector concatenation.

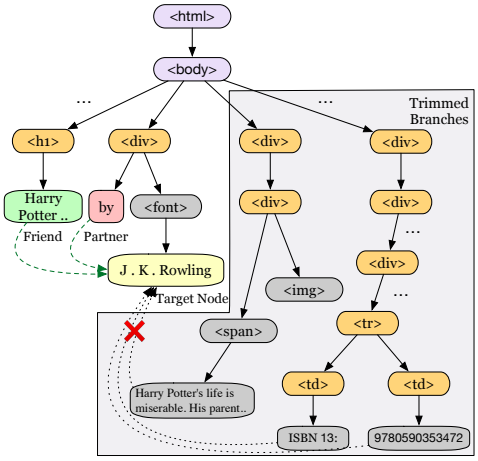

Figure 4: The DOM Tree Simplification Module extracts the partner (by) and friends (Harry Potter and the Sorcerer's Stone) for each node (J. K. Rowling) by trimming unrelated branches.

the DOM trees to extract context features for each node. All the textual features are then fed into a text encoder to generate a dense semantic embedding. We find that extra discrete features built from markup information such as XPath and leaf node type can result in a better node representation. We also add the relative position of each node as a global feature for the extraction task. The combined node embedding is used for predicting the type of node. In the intra-vertical scenario, we directly apply a multi-class classifier to the node embedding and output the attribute type probability distribution. In the cross-vertical scenario, the attribute sets differ from vertical to vertical. Therefore, we have to alter the inference strategy to binary classification to achieve a matching probability for each attribute type. Then, we select the attribute with the highest probability as the prediction.

3 NODE ENCODER AND CLASSIFIER

The node encoder consists of three components: DOM tree simplification module, text encoder, and discrete feature module.

3.1 DOM Tree Simplification Module

In this module, we simplify the DOM tree to extract the contexts for each variable node x, namely its friend circle features, which are composed of the node's *partner* and *friends*. The whole DOM tree is a collection of nodes that originate from a unique starting node called the *root*. The set of nodes A on the path from *root* to node x (not including x) are ancestors of node x. The *friends* of x denotes a set of text nodes $X^F \subseteq X \cup Y \backslash \{x\}$. For each $x^f \in X^F$, the distances from both x^f and x to their lowest common ancestor $a \in A$ should be no more than constant N. We compute the distance by counting the number of edges on the path. The *partner* x^p of x is a special *friend* node for which x and x^p are the only two text nodes in the tree that originates from their lowest common ancestor. Note that each node has at most one *partner* in the DOM tree while it could have zero or multiple *friends*. Usually, *partner* x^p is the closest *friend* to x in the DOM tree.

Algorithm 1: Function $\mathcal{F}$ for DOM Tree Simplification and Friend Circle Extraction.

Input: DOM tree T variable node set X constant K;
Output: Dictionaries D_p and D_f where each key is $x \in X$ and the values are its corresponding partner and friend set, respectively;
Initialize D_d, D_p, D_f as three empty dictionaries;
for *each variable node* $x \in X$ **do**
  Get the node's XPath P_x from T;
  Generate the node's ancestor set ANC_x from P_x and mark the ordered closest K ancestors as ANC_x^K;
  for *each anc in* ANC_x^K **do**
    Add x to $D_d[anc]$;
  end
end
for *each variable node* $x \in X$ **do**
  for *each anc* $\in ANC_x^K$ **do**
    $DESC \leftarrow D_d[anc] \backslash \{x\}$;
    if *there exists only one node* x' *in DESC and both* $D_p[x], D_f[x]$ *are empty* **then**
      Add x' to $D_p[x]$;
    $D_f[x] \leftarrow D_f[x] \cup DESC$;
  end
end

As function $\mathcal{F}$ described in Algorithm 1, for each variable node $x \in X$, we decode its XPath information to record the K closest ancestors of x. For instance, if the XPath of x is "/body/tr/td/", we consider both "/body/tr/" and "/body/" as the ancestor of x. Reversely, we can easily obtain all the descendants of each ancestor node to composite the candidate set for retrieving the partner and friends. By limiting the size of K, we can narrow down the search area in the tree such that the noisy textual features from distant branches can be efficiently trimmed, as shown in Figure 4.

In the extraction process, we keep all the basic HTML element tags like <tr> and <td> while remove the formatting and style tags such as and <font>[2]. In Figure 5, we plot a common sub tree structure (a) and its three possible variants (b,c,d). With Algorithm 1, we can simplify and normalize the three variants to (a) in order to extract the friend circle features.

With partner and friends extracted from the DOM tree for each node x, we feed the three sets of textual features separately into the text encoder as described in section 3.2 to generate three representations e_x, e_p, and e_f which are all d_w-dimensional vectors. We derive the joint semantic embedding e_s by simply concatenating the three representations as follows:

$$e_s = \left[e_x; e_p; e_f\right].$$

Note that the joint embedding is a $3d_w$-dimensional vector.

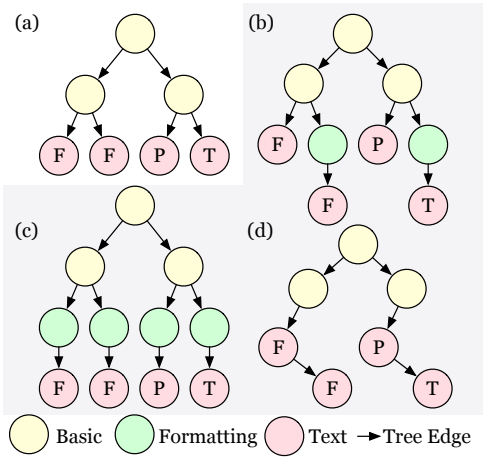

Figure 5: Subtree skeletons of web page DOMs including a common structure (a) and its three possible variants (b), (c) and (d). "Basic" denotes a set of basic HTML element tags, while "Formatting" represents some formatting and style tags such as and <font>. The "Text" node has text information. We aim to simplify all possible variants to (a), in order to efficiently extract the partner (marked as P) and friends (F) for each target node (T).

3.2 Text Encoder

Node x contains a sequence of text $S_1 = [w_1, w_2, ..., w_{L1}]$, where $w_i \in \mathcal{W}$ and $L1$ denotes the word sequence length. We can easily split each word into a sequence of characters $S_2 = [c_1, c_2, ..., c_{L2}]$, where $c_i \in C$ and $L2$ is the character sequence length. $\mathcal{W}$ and C are vocabularies of words and characters. We employ a hierarchical LSTM-CNN text encoder to encode the character-level and word-level features.

We notice that the attribute values usually contain useful morphological patterns in the character-level semantics [21]. For example, (*aa'bb* ft) and (*aa-bb* ft) are two common patterns of *height* attribute in the *nbaplayer* vertical. Their character-level representation can be very essential. Therefore, we leverage a Convolutional Neural Network to encode the character-level embeddings (dimension d_c) of each word w, resulting in h^c_w. We simply concatenate h^c_w with its word-level representation g_w retrieved from external pretrained word embeddings: $h_w = \left[g_w; h^c_w\right]$.

The LSTM [16] has been widely used as the unit of Recurrent Neural Network for learning the latent representation of sequence data [23]. Therefore, we feed the latent word representations $\left[h_{w_1}, h_{w_2}, ..., h_{w_{L1}}\right]$ into a bi-directional LSTM network, resulting in $e_x = \left[h^{forward}_w; h^{backward}_w\right]$.

Similarly, we can achieve the semantic representations for the node's partner and friends, e_p and e_f.

3.3 Discrete Feature Module

Xpath embeddings. Markup features such as XPath can be very useful for node tagging. An XPath of a DOM node "/html/body/tr/td/" can be seen as a sequence of HTML tags [<html>, <body>, <tr>, <td>]. We learn a separate bi-directional LSTM to get the dense representation e_{xpath} of dimension d_{xpath} for each XPath sequence such that it can make use of all the meaningful tags in the sequence.
Leaf node type embeddings. The tag type of the DOM leaf node such as "<h1>" can also be meaningful. "<h1>" means the node is likely to be the title of the page, highly correlating with the *name* of a *nbaplayer* or the *title* of a *book*. We collect the vocabulary set of the HTML tags and randomly initialize an embedding e_{leaf} of dimension d_{leaf} for each of them.
Position embeddings. We also leverage the relative position of each node x as a discrete feature. This global information can benefit the task. For example in the *auto* vertical, the *model* usually lies on the top of the page. We apply depth-first-search to traverse the tree and get the occurrence position pos_x of each node. Then we compute its relative position via $\lceil \frac{pos_x}{\max_x\{pos_x\}} \rceil$. Similarly, a random embedding e_{pos} of dimension d_{pos} is initialized for each position.
Semantic similarity. We notice the for each node x the text in the partner node x^p can help determine x's attribute type and modeling the semantic relation between the text in x^p and the attribute types allows us to best leverage this data. Specifically, we compute the *cosine similarity*[3] between the partner embedding e_p and each attribute embedding e_{a_i} to model their semantic relations, which results in a semantic similarity vector e_{cos} of dimension M, where M denotes the number of pre-defined attribute types.

Upon achieving these discrete features, we concatenate them into a vector $e_d = \left[e_{xpath}; e_{leaf}; e_{pos}; e_{cos}\right]$ of dimension d_{xpath}+d_{leaf}+d_{pos}+M.

[2]We refer to the HTML tag categories in https://www.w3schools.com/TAGS/ref_byfunc.asp.

[3]We compute the scores via *cosine_similarity* $(e_p, e_{a_i}) = \frac{e_p \cdot e_{a_i}}{|e_p||e_{a_i}|}$.

3.4 Inference and Optimization

Under the intra-vertical scenario, the node embedding is connected to a multi-layer perceptron (MLP) for multi-class classification, as illustrated below:

$$e_n = [e_s; e_d]$$
$$\mathbf{h} = \text{MLP}(e_n), \mathbf{h} \in \mathbb{R}^{M+1}.$$

where $M+1$ denotes the number of pre-defined attribute types plus a *none* type.

Under the cross-vertical scenario, we notice each vertical has a different attribute set. The MLP layer for multi-class classification can no longer be reused for different verticals which have different sizes of attribute sets. Therefore, we alter the inference strategy to binary classification. We individually concatenate the node embedding e_n to each attribute embedding e_{a_i} of dimension d_a which is randomly initialized. We then connect it to a separate MLP and compute a score $\mathbf{h}_i$ for each attribute type:

$$e_{b_i} = \left[e_n; e_{a_i}\right], 1 \le i \le M+1$$
$$\mathbf{h}_i = \text{MLP}(e_{b_i}), \mathbf{h}_i \in \mathbb{R}$$

Under both scenarios, we lastly apply the *softmax* function to normalize $\mathbf{h}$ and select the largest as the prediction $\hat{\mathbf{y}}$:

$$\mathbf{p}_i = \frac{e^{\mathbf{h}_i}}{\sum_{j=1}^{M+1} e^{\mathbf{h}_j}}; \hat{\mathbf{y}} = \arg\max_i \mathbf{p}_i.$$

The loss function optimizes the cross-entropy between the true labels $\mathbf{y}$ and the normalized probabilistic scores $\mathbf{p}$.

$$\text{loss} = -\sum_{n=1}^{|X|}\sum_{m=1}^{M+1} \mathbf{y}_{m,n} \log \mathbf{p}_{m,n}$$

4 EXPERIMENTS

In this section, we firstly introduce the dataset and evaluation metrics. We also explain the implementation details to guarantee the reproducibility of our method. Then, a collection of baseline models are introduced to compare with our model under the intra-vertical few-shot extraction scenario. We also conduct a series of ablation studies to answer the following questions: (i) *What are the contributions from each set of features?* (ii) *Will sequence modeling work well on DOM tree nodes?* (iii) *What are the performances of using different word embedding strategies?* Lastly, we evaluate the effectiveness of the out-of-domain knowledge under the cross-vertical few-shot extraction scenario.

4.1 Dataset

We rely on a public data set, SWDE [15] that consists of more than 124,000 web pages from 80 websites of 8 verticals to train and evaluate the proposed model. Detailed statistics are shown in Table 1. Each vertical consists of 10 websites and contains 3 to 5 attributes of interest. We notice *book* and *job* have the most variable nodes on average which is roughly three times the nodes in vertical *auto* and *university*.

In the intra-vertical few-shot experiments, we follow the settings in FreeDOM [21] to randomly select k seed websites as the training data and use the remaining $10-k$ websites as the test set. Note that in this few-shot extraction task, none of the pages in the $10-k$ websites have been visited in the training phase. This setting is abstracted from the real application scenario where only a small set of labeled data is provided for specific websites and we aim to infer the attributes on a much larger unseen website set.

Vertical	#Sites	#Pages	#Var. Nodes	Attributes
auto	10	17,923	130.1	model, price, engine, fuel
book	10	20,000	476.8	title, author, isbn13, pub, date
camera	10	5,258	351.8	model, price, manufacturer
job	10	20,000	374.7	title, company, location, date
movie	10	20,000	284.6	title, director, genre, mpaa
nbaplayer	10	4,405	321.5	name, team, height, weight
restaurant	10	20,000	267.4	name, address, phone, cuisine
university	10	16,705	186.2	name, phone, website, type

Table 1: SDWE Dataset Statistics

In the cross-vertical few-shot experiments, we leverage one vertical as the out-of-domain knowledge to train a model. Then we conduct the same intra-vertical extraction experiments by loading the checkpoints from the pretrained model for parameter initialization. We create this experimental setting to enable a broader knowledge transfer across various verticals, which can tackle the scenario where the domain of the existing annotation is inconsistent with the unseen websites.

4.2 Evaluation Metrics

We evaluate the extraction performance by page-level F1 scores, following the evaluation metrics from SWDE and FreeDOM [15, 21]. Page-level F1 score is the harmonic mean of extraction precision and recall in each page. Specifically, we evaluate the predicted attribute values with the true values for each detailed page. We compute an average F1 score over all the verticals (Table 2) to compare with the baselines. We also compute the average F1 score for each vertical (Figure 6) and each attribute (Figure 7) for detailed analysis.

4.3 Implementation details

For data pre-processing, we use open-source LXML library[4] to process each page for obtaining the DOM tree structures. Then, we follow the simple heuristic used in [21] to filter nodes whose values are constant in all pages of a website, thus most of the noisy page-invariant textual nodes such as the footer and navigation contents are removed and the experiments are significantly accelerated in terms of the training speed. We use GloVe pretrained representations [29] to initialize our word embeddings. Other representations such as character embeddings and attribute embeddings are all randomly initialized. We also cut off every node's text when it has more than 15 words. We set both maximum edge number N and maximum ancestor number K as 5 for extracting friend circle features and only keep the closest 10 friends for each DOM tree node by comparing their relative positions on the web page.

We conduct a grid search for all the hyper-parameters. We use 100 for both word embedding size d_w and character embedding size d_c. We select d_{path}, d_{leaf}, d_{pos} as 30, 30, 20, respectively. For the CNN network, we use 50 filters and 3 as kernel size. For the LSTM network, we set the hidden layer size as 100. The model is implemented in Tensorflow. We train the model with epoch number 15 and a batch size 32. We apply a dropout mechanism following

[4] https://lxml.de/

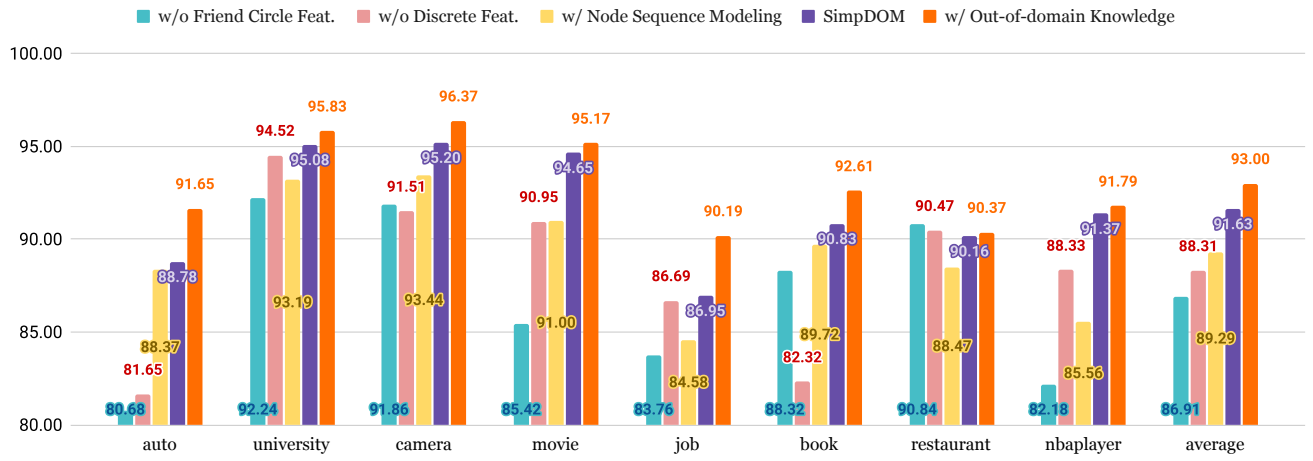

Figure 6: Ablation study results that demonstrate the contribution from different features and modules. We conclude that both friend circle features and discrete features improve the extraction performance while adding a sequence modeling module harms the performance dramatically. With the out-of-domain knowledge from a second vertical, the model can do better for each of them. We set $k = 3$ here. Similar results can be achieved with other k's.

the MLP layer to avoid over-fitting issues. The dropout rate is 0.3. We use Adam as the optimizer where the learning rate is 0.001. It takes less than 30 minutes to finish the a complete training and evaluation cycle for each vertical with one NVIDIA V100 GPU.

4.4 Baseline Models

We compare against several baselines:

Stacked Skews Model (SSM). SSM [5] utilizes expensive hand-crafted features and tree alignment algorithms to align the unseen web pages with seed web pages. This method does not require visual rendering features, which is the same as our model.

Rendering-feature Model (Render-full). Render-full [15] employs visual features to express the distances between node blocks rendered with the web browser. Visual distances are proven a good feature to encode the neighboring relationships among nodes [25] but this method requires the time-consuming rendering process and needs extra memory space to save the images, CSS, and JavaScripts that can easily be out-of-date. In specific, Render-full employs a sophisticated heuristic algorithm to compute the visual distances, which gives the best performance [15], compared to other variants Render-PL and Render-IP.

Relational Neural Model (FreeDOM-X). FreeDOM leverages a relational neural network to encode features such as the relative distance and text semantics. This method is composed of two stages. The first stage model (FreeDOM-NL) learns a dense representation for each DOM tree node via node-level classification. The relational neural network in the second stage (FreeDOM-Full) claims to capture the distance and semantic relatedness between pairs of nodes in the DOM trees. This two-stage model does not rely on visual features but is hard to be deployed in practice. Besides, only modeling the relatedness between pairs of nodes neglects the rich structural information in the tree such as the friend circles. We compare with both FreeDOM-NL and FreeDOM-Full because the single-stage FreeDOM-NL is closer to our model and FreeDOM-Full achieves the state-of-the-art experimental results.

Model \ #Seed Sites	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$
SSM	63.00	64.50	69.20	71.90	74.10
Render-Full	**84.30**	86.00	86.80	88.40	88.60
FreeDOM-NL	72.52	81.33	86.44	88.55	90.28
FreeDOM-Full	82.32	86.36	90.49	91.29	92.56
SimpDOM	83.06	**88.96**	**91.63**	**92.84**	**93.75**

Table 2: Comparing the extraction performance (F1 score) of five baseline models to our method SimpDOM using different numbers of seed sites $k = \{1, 2, 3, 4, 5\}$. Each value in the table is computed from the average over 8 verticals and 10 permutations of seed websites per vertical (80 experiments in total).

4.5 Intra-vertical Few-shot Extraction Results

Table 2 shows the overall comparisons between our model SimpDOM and all four baselines using different numbers of seed websites. Our model achieves a slightly worse performance when $k = 1$ while largely outperforms Render-Full when $k = \{2, 3, 4, 5\}$. We can conclude that the delicately crafted visual features can capture more patterns in the scenario where extremely small training data exists. However, they are not as transferable as the rich semantic features extracted from our simplified DOM trees as k increases. Our method also consistently outperforms the state-of-the-art method FreeDOM-Full (an average lift of 1.44% over all the k's) and achieves a 3.47%-10.54% improvement from the single-stage approach, FreeDOM-NL, per F1 score.

We plot the detailed performance of SimpDOM on different verticals in figure 8. In general, the performance is improved as k

Figure 7: Per-attribute F1 performance comparisons between SimpDOM w/ and w/o friend circle features. We set $k = 3$ here. Attributes like *height* in *nbaplayer* and *title* in *movie* get the largest performance lifts.

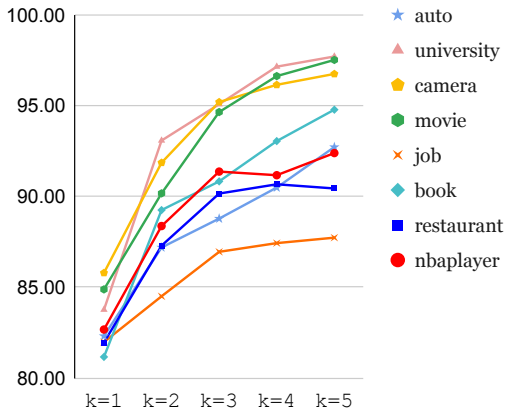

Figure 8: Comparing the extraction performance (F1 score) of different numbers of seed sites $k = \{1, 2, 3, 4, 5\}$ per vertical.

increases. This is not surprising because more training data obtain better coverage of all possible instances. we also observe that the rate of performance growth slows down and sometimes the F1 scores of some verticals (e.g. *nbaplayer* and *restaurant*) even fluctuates as more data join the training process (i.e. as k increases). We think the reason is that the model becomes more robust and less new knowledge can be transferred from annotated websites to unseen websites in these verticals.

4.6 Ablation Study

In Figure 6, we demonstrate an ablation study on different features of SimpDOM, including discrete features and friend circle features. We find that both sets of features improve the attribute extraction performance dramatically. For instance, the friend circle features lift up the F1 score of *nbaplayer* vertical from 82.18% to 91.37% and the discrete features increase the performance on *book* vertical by 8.51%. However, *restaurant* is a special case where the result drops when we employ either of the two feature sets. We believe the node texts in some attribute values such as *name* and *address* are distinguishable enough and adding more features just brings more noise to the classification. This is also corroborated by Figure 7, which explains the detailed performance change when adding the friend circle features per attribute. We observe that the improvement on *height* of *nbaplayer* is significant. The nodes containing *height* value always share a similar pattern xx-yy[5] with some other nodes on the same page. With the friend circle features, we find that *weight* is always a friend node of *height*, which makes *height* distinguishable from other nodes with similar text patterns.

Another interesting ablation study is done with an additional sequence modeling layer[6] which is commonly applied to sequence labeling tasks such as named entity recognition on plain text [20, 41]. We first obtain a sequence of node embeddings before the MLP classifier where all the nodes are from one web page. Then a new representation can be achieved from the sequence model for each node. The same classifier is used to predict the attribute type with the updated node representation. As shown in Figure 6 (marked as "w/ Node Sequence Modeling"), the additional sequence modeling layer fails to optimize the node representations for all the verticals especially those with more variable nodes such as *nbaplayer* and *job*. We suppose that the information from all other DOM tree nodes can be selectively attended to the current node with such mechanism, which however introduces more noise than useful knowledge. This further proves the importance of utilizing the

[5] For instance, NBA player Kobe Bryant's height (6-6) has the same value as his shooting record (6-6) in one game. It is impossible to distinguish two nodes by the text.
[6] We utilize the Transformer [38] as the sequence modeling layer. LSTM can be an alternative.

structures in the simplified DOM trees to eliminate the noise from distant and irrelevant nodes.

Embedding Approach	F1	Performance Change
GloVe Embedding Trainable	**91.63**	0
GloVe Embedding Fixed	91.25	-0.38
Randomized Word Embedding	89.66	-1.97
Contextualied Embedding	81.83	-9.80

Table 3: Comparing different word embedding approaches when $k = 3$.

We also compare the different embedding approaches for encoding textual features. As shown in Table 3, we conduct experiments to test the randomized word embedding, fixed GloVe word embedding, and trainable GloVe word embedding. In the trainable setting, we can continue to optimize the parameters in the embedding layer which is initialized from GloVe and it gets the best performance. We think a specific "web-language" model can serve the web information extraction tasks better. As contextualized language models develop nowadays, we also try the BERT [11][7] to generate the contextualized embeddings but it decreases the performance by 9.8%. It is not surprising because the context in each node is very limited[8] and the huge size of parameters (110M in BERT-BASE) for fine-tuning can easily cause an over-fitting problem.

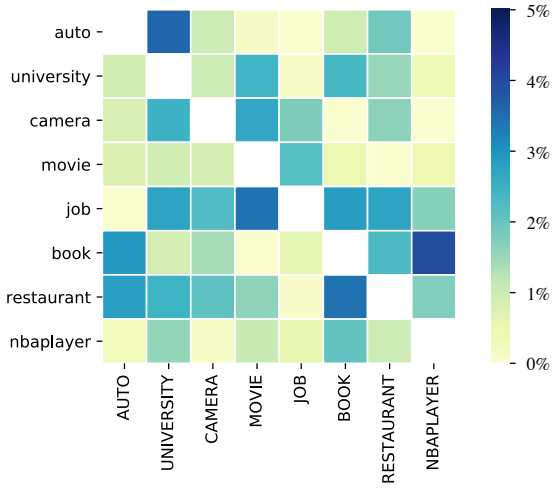

Figure 9: Heatmap denoting the performance lifts per F1 score from the out-of-domain knowledge. In specific, we learn a transferable model with verticals in upper case (columns). Then we finetune the model and predict on the verticals in lower case (rows).

4.7 Cross-vertical Few-shot Extraction Results

We plot a heatmap in Figure 9 to denote the performance lifts from the out-of-domain knowledge. In specific, each entry in the heatmap relates to a pair of verticals, where the vertical in the upper case is used as the out-of-domain knowledge while the vertical in the lower case is used to train and test the model. We do not plot the scores in the diagonal because every vertical cannot serve as its out-of-domain resource. One interesting observation is that this heatmap is roughly symmetric with respect to the diagonal, which demonstrates a mutual relationship between pairs of verticals. For instance, *job* and *movie*, *book* and *nbaplayer*, *restaurant* and *book* can all significantly improve the extraction performance for each other, while *auto* and *job*, *camera* and *nbaplayer* seem to be irrelevant to each other. We show the performance of each vertical achieved by using the most helpful vertical's out-of-domain knowledge in Figure 6. We achieve the highest average F1 score 93% over all the verticals ($k = 3$).

5 RELATED WORK

5.1 Web Information Extraction

Web information extraction processes vast amount of unstructured or semi-structured contents from the web and has drawn a lot of attention from the data mining research community [3, 6, 13, 22, 31]. Four broad categories of web information extraction tasks can be summarized. They are attribute (entity) extraction, relation extraction, composite extraction, and application-driven extraction.

Attribute extraction targets to identify named entity mentions such as book price, phone number, movie title from web documents. Though this task is intuitive to describe, the high-quality corpus annotation requires time-consuming human-crafted rules and dictionaries [5, 15, 21, 28].

Relation extraction associates pairs of named entities and identifies a pre-defined relationship between them. Closed relation extraction defines a closed set of relation types including a special type indicating "no relation" while open relation extraction conducts a binary classification of whether there exists a relationship between the two entities [1, 24, 25, 32, 46].

Composite extraction aims to extract more complex concepts such as reviews, opinions, and sentiment mentions. Attribute and relation extractions can be integrated into the high-level workflow of composite extraction with other sub-modules like sentiment classification or aspect detection [8–10, 34, 36].

Application-driven extraction includes a broad spectrum of application scenarios such as web representation learning, PDF information extraction using OCR techniques, anomaly detection of web-based attacks and so on [17, 18, 26, 33, 37, 45].

5.2 Attribute Extraction from Web Documents

Attribute extraction serves as the fundamental task in the web information extraction pipelines and enables a wide range of downstream applications [4, 12, 39, 40]. However, there still exists a huge room to develop attribute extraction methods of high accuracy and strong transferability. Traditional approaches [2, 7, 19, 27, 35, 43, 44] either reply on analyzing the templates that are used to build the web pages or leverage unsupervised models to tackle the problem. However, they neglect the rich semantics of the attribute values and require considerable human efforts for annotations, thus failing to be generalizable to unseen websites. Some recent methods [5, 15] believe utilizing visual features generated from the web page rendering process can enable the model to extract attributes from

[7] We choose BERT without loss of generality. It can be replaced by its alternatives like ELMo [30] or XLNet [42].
[8] On average, each variable node contains only 2-5 words in different verticals.

new websites. Nevertheless, it is time-consuming to build visual features and space-unfriendly to store the necessary images, CSS, JavaScript files that are prone to be out-of-date. In this paper, we aim to construct a transferable model to extract attributes from unseen websites without using any visual features.

6 CONCLUSION

In this paper, we propose a simple but effective method, SimpDOM, that simplifies the DOM trees to extract informative and transferable knowledge for the attribute extraction task. We build a rich representation for each DOM tree node without using any visual features. Extensive experiments show that SimpDOM significantly outperforms the SOTA method by 1.44% on the F1 score and utilizing out-of-domain knowledge further improves the performance by 1.37%. We will open-source the implementations to facilitate further researches in the web data mining community.

REFERENCES

[1] I. Augenstein, D. Maynard, and F. Ciravegna. Distantly supervised web relation extraction for knowledge base population. *Semantic Web*, 7(4):335–349, 2016.

[2] M. A. B. M. Azir and K. B. Ahmad. Wrapper approaches for web data extraction: A review. In *2017 6th International Conference on Electrical Engineering and Informatics (ICEEI)*, pages 1–6. IEEE, 2017.

[3] R. Baumgartner, S. Flesca, and G. Gottlob. Visual web information extraction with lixto. 2001.

[4] L. Bing, T.-L. Wong, and W. Lam. Unsupervised extraction of popular product attributes from e-commerce web sites by considering customer reviews. *ACM Transactions on Internet Technology (TOIT)*, 16(2):1–17, 2016.

[5] A. Carlson and C. Schafer. Bootstrapping information extraction from semi-structured web pages. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 195–210. Springer, 2008.

[6] C.-H. Chang, M. Kayed, M. R. Girgis, and K. F. Shaalan. A survey of web information extraction systems. *IEEE transactions on knowledge and data engineering*, 18(10):1411–1428, 2006.

[7] C.-H. Chang and S.-C. Lui. Iepad: information extraction based on pattern discovery. In *Proceedings of the 10th international conference on World Wide Web*, pages 681–688, 2001.

[8] W. Chen, L. Zong, W. Huang, G. Ou, Y. Wang, and D. Yang. An empirical study of massively parallel bayesian networks learning for sentiment extraction from unstructured text. In *Asia-Pacific Web Conference*, pages 424–435. Springer, 2011.

[9] S. R. Das and M. Y. Chen. Yahoo! for amazon: Sentiment extraction from small talk on the web. *Management science*, 53(9):1375–1388, 2007.

[10] K. Dave, S. Lawrence, and D. M. Pennock. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *Proceedings of the 12th international conference on World Wide Web*, pages 519–528, 2003.

[11] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[12] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 601–610, 2014.

[13] O. Etzioni, M. Banko, S. Soderland, and D. S. Weld. Open information extraction from the web. *Communications of the ACM*, 51(12):68–74, 2008.

[14] S. Gupta, G. Kaiser, D. Neistadt, and P. Grimm. Dom-based content extraction of html documents. In *Proceedings of the 12th international conference on World Wide Web*, pages 207–214, 2003.

[15] Q. Hao, R. Cai, Y. Pang, and L. Zhang. 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*, pages 775–784, 2011.

[16] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[17] F. Kocayusufoglu, Y. Sheng, N. Vo, J. Wendt, Q. Zhao, S. Tata, and M. Najork. Riser: Learning better representations for richly structured emails. In *The World Wide Web Conference*, pages 886–895, 2019.

[18] C. Kruegel, G. Vigna, and W. Robertson. A multi-model approach to the detection of web-based attacks. *Computer Networks*, 48(5):717–738, 2005.

[19] N. Kushmerick, D. S. Weld, and R. Doorenbos. *Wrapper induction for information extraction*. University of Washington Washington, 1997.

[20] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*, 2016.

[21] B. Y. Lin, Y. Sheng, N. Vo, and S. Tata. Freedom: A transferable neural architecture for structured information extraction on web documents. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1092–1102, 2020.

[22] M. T. Ling Liu. *Encyclopedia of Database Systems*. Springer New York, 2nd ed. edition, 2018.

[23] P. Liu, X. Qiu, and X. Huang. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101*, 2016.

[24] C. Lockard, X. L. Dong, A. Einolghozati, and P. Shiralkar. Ceres: Distantly supervised relation extraction from the semi-structured web. *arXiv preprint arXiv:1804.04635*, 2018.

[25] C. Lockard, P. Shiralkar, X. L. Dong, and H. Hajishirzi. Zeroshotceres: Zero-shot relation extraction from semi-structured webpages. *arXiv preprint arXiv:2005.07105*, 2020.

[26] B. P. Majumder, N. Potti, S. Tata, J. B. Wendt, Q. Zhao, and M. Najork. Representation learning for information extraction from form-like documents. In *proceedings of the 58th annual meeting of the Association for Computational Linguistics*, pages 6495–6504, 2020.

[27] I. Muslea, S. Minton, and C. Knoblock. A hierarchical approach to wrapper induction. In *Proceedings of the third annual conference on Autonomous Agents*, pages 190–197, 1999.

[28] P. Pasupat and P. Liang. Zero-shot entity extraction from web pages. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 391–401, 2014.

[29] J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *EMNLP 2014*, pages 1532–1543, 2014.

[30] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. In *NAACL 2018*, pages 2227–2237, 2018.

[31] B. Popov, A. Kiryakov, D. Ognyanoff, D. Manov, A. Kirilov, and M. Goranov. Towards semantic web information extraction. In *Human Language Technologies Workshop at the 2nd International Semantic Web Conference (ISWC2003)*, volume 20, 2003.

[32] C. Quirk and H. Poon. Distant supervision for relation extraction beyond the sentence boundary. *arXiv preprint arXiv:1609.04873*, 2016.

[33] C. Ramakrishnan, A. Patnia, E. Hovy, and G. A. Burns. Layout-aware text extraction from full-text pdf of scientific articles. *Source code for biology and medicine*, 7(1):7, 2012.

[34] S. K. Shandilya and S. Jain. Automatic opinion extraction from web documents. In *2009 International Conference on Computer and Automation Engineering*, pages 351–355. IEEE, 2009.

[35] S. Soderland. Learning information extraction rules for semi-structured and free text. *Machine learning*, 34(1-3):233–272, 1999.

[36] X. Song, J. Liu, Y. Cao, C.-Y. Lin, and H.-W. Hon. Automatic extraction of web data records containing user-generated content. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 39–48, 2010.

[37] A. M. Vartouni, S. S. Kashi, and M. Teshnehlab. An anomaly detection method to detect web attacks using stacked auto-encoder. In *2018 6th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS)*, pages 131–134. IEEE, 2018.

[38] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[39] H. Wang, F. Zhang, M. Zhao, W. Li, X. Xie, and M. Guo. Multi-task feature learning for knowledge graph enhanced recommendation. In *The World Wide Web Conference*, pages 2000–2010, 2019.

[40] S. Wu, L. Hsiao, X. Cheng, B. Hancock, T. Rekatsinas, P. Levis, and C. Ré. Fonduer: Knowledge base construction from richly formatted data. In *Proceedings of the 2018 International Conference on Management of Data*, pages 1301–1316, 2018.

[41] H. Yan, B. Deng, X. Li, and X. Qiu. Tener: Adapting transformer encoder for name entity recognition. *arXiv preprint arXiv:1911.04474*, 2019.

[42] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*, 2019.

[43] Y. Zhai and B. Liu. Web data extraction based on partial tree alignment. In *Proceedings of the 14th international conference on World Wide Web*, pages 76–85, 2005.

[44] S. Zheng, R. Song, J.-R. Wen, and D. Wu. Joint optimization of wrapper generation and template detection. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 894–902, 2007.

[45] Y. Zhou, J.-Y. Jiang, K.-W. Chang, and W. Wang. Learning to discriminate perturbations for blocking adversarial attacks in text classification. *arXiv preprint arXiv:1909.03084*, 2019.

[46] A. Zouaq, M. Gagnon, and L. Jean-Louis. An assessment of open relation extraction systems for the semantic web. *Information Systems*, 71:228–239, 2017.