Learning Transferable Node Representations for Attribute Extraction from Web Documents

Yichao Zhou, Ying Sheng, Nguyen Vo, Nick Edmonds, Sandeep Tata
Google
Mountain View, USA
{yichaojoey,yingsheng,nguyenvo,nge,tata}@google.com

ABSTRACT

Given a web page, extracting an 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. Prior approaches have either relied on computationally expensive visual feature engineering or required large amounts of training data to get to an acceptable precision. In this paper, we propose a novel method, LeArNing TransfErable node RepresentatioNs for Attribute Extraction (LANTERN), to tackle the problem. We model the problem as a tree node tagging task. The key insight is to learn a contextual representation for each node in the DOM tree where the context explicitly takes into account the tree structure of the neighborhood around the node. Experiments on the SWDE public dataset show that LANTERN outperforms the previous state-of-the-art (SOTA) by 1.44% (F1 score) with a dramatically simpler model architecture. Furthermore, we report that utilizing data from a different domain (for instance, using training data about web pages with cars to extract book objects) 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:
Yichao Zhou, Ying Sheng, Nguyen Vo, Nick Edmonds, Sandeep Tata. 2022. Learning Transferable Node Representations for Attribute Extraction from Web Documents. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining (WSDM '22), February 21–25, 2022, Tempe, AZ, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3488560.3498424

1 INTRODUCTION

The World Wide Web contains vast amounts of information in a semi-structured format. Translating this information into structured knowledge has long been an important research goal [6, 14].

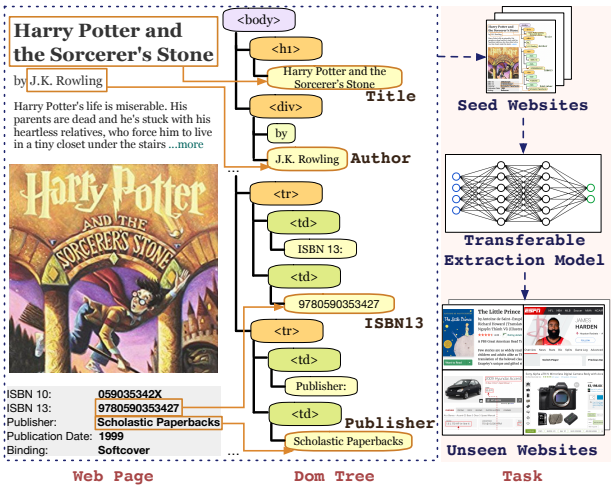

Figure 1: Learning a transferable model based on HTML DOM trees to extract attributes from unseen websites of various domains. Note that every web page is rendered from a Document Object Model (DOM) Tree. All the attributes are located in the leaf nodes of the trees.

Extracting structured objects with relevant attributes from semi-structured HTML can power applications including large-scale knowledge base/graph construction [10, 42], e-commerce product search [3, 14], and personalized recommendation [41]. Attribute extraction from web pages is complicated by the semi-structured data format, noisy page contents, complex formatting, and imperfect alignment of the source and visual representations. Whereas unstructured texts can easily be modeled as a sequence [25], web pages demand more sophisticated techniques.

In this paper, we focus on the problem of extracting structured objects with a given target schema (like a book, such as in Figure 1, to extract attributes of interest like {*title, author, isbn13, publisher*}) using a small amount of labeled data (e.g. a few websites). We consider two challenging scenarios in this paper, (i) intra-domain few-shot extraction, where the training data consists of a few labeled seed websites from a given domain and the task is to extract the structured object from unseen websites in the same domain; (ii) cross-domain few-shot extraction, where the training data consists of a few labeled seed websites from a given domain (say *A*) *and* additional labeled websites from a *different* domain (say *B*) and the task is to extract structured objects from unseen websites in domain *A*. The key difference in the cross-domain setting is the availability of additional labeled websites from a *different* domain. At first glance, one may wonder why training data about one domain (say books) might help an extraction model on a completely unrelated domain

WSDM '22, February 21–25, 2022, Tempe, AZ, USA

ACM ISBN 978-1-4503-9132-0/22/02.
https://doi.org/10.1145/3488560.3498424

(say cars). The experimental evidence in this paper suggests that this is indeed helpful. We believe this is because the model is able to take advantage of this additional data to learn transferable contextual representations for the nodes. Practically, this is of significant value: the task of building each successive extractor is made easier by leveraging the websites labeled for previous tasks.

Our proposed LANTERN model builds a rich representation for each node in the DOM tree by focusing on contextual features. This representation is then used to train a classifier to decide which attribute type the node belongs to. A key insight in this paper is an algorithm to simplify the DOM trees and identify "friend" and "partner" nodes that are a particularly valuable context signal. 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. Knowing that the node containing "by" is a critical part of the contextual clue a human rely on to determine that "J. K. Rowling" is the author of this book.

Summarizing node contexts using "friend" and "partner" simplifies the inputs required to compute the node representation, and allows us to leverage website-invariant features (e.g. semantically informative expressions) and domain-invariant clues (e.g. the co-occurrence of multiple attribute values). Nodes that contain attribute values of interest are often close to one another, reminding us to pay special attention to the neighbor relationship of DOM nodes. Our contributions are as follows:

- To the best of our knowledge, this is the first work to learn contextual representations for DOM tree nodes in a web page by leveraging the local tree structure.
- We are the first to present the cross-domain, few-shot attribute extraction task and demonstrate the improved performance compared to the intra-domain setting.
- Extensive experiments show that LANTERN significantly outperforms the SOTA by 1.44% (F1 score), and the out-of-domain knowledge helps beat the SOTA by a further 1.37%.
- We open-sourced our implementations at https://github.com/google-research/google-research/tree/master/simpdom to provide a testbed for future research in this direction.

There's a rich history of related work in this space which we address in the next section. In particular, we distinguish our work from the literature on wrapper induction [2, 19, 29] which relies on the fact that many websites are created from Document Object Model (DOM) tree [13] templates. These techniques have two drawbacks: 1) They typically require a labeled example for each site in the target domain to induce a wrapper for other pages on that site, 2) The wrappers yield work well for exact copies of the DOM structure but can be brittle in the face of minor structural variation or web page evolution over time [30]. Thus considerable human effort is required to periodically update templates. Our approach, LANTERN eschews wrapper induction and learns attribute extraction models from a limited amount of annotated data that are capable of generalizing to web sites *not present* in the training data.

2 RELATED WORK

Web information extraction processes a vast amount of unstructured or semi-structured contents from the web and has drawn a lot of attention from the data mining research community [6, 11, 22]. Four broad categories of web information extraction tasks can be described: attribute (entity) extraction, relation extraction, composite extraction, and application-driven extraction. Attribute extraction attempts 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, 14, 21, 31]. 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, 25, 26]. Composite extraction aims to extract more complex concepts such as reviews, opinions, and sentiment mentions [7, 35, 38], while 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 [18, 27, 39].

Attribute extraction is a fundamental task in web information extraction pipelines and enables a wide range of downstream applications [3, 41]. Ideally, attribute extraction methods should have both high accuracy and strong transferability to new domains.

Traditional approaches [2, 4, 8, 12, 16, 19, 29, 36, 37, 45] have long been based on **wrappers** which depend on the alignment of DOM-tree based templates that are used to generate web pages. These methods are brittle in the face of variation across pages or evolution over time, requiring considerable human efforts to periodically update templates and annotate new pages. Having a wrapper for one domain contributes little to the task of building a wrapper for another domain. For instance, FiVaTech [16] can only generate the template for each website separately by comparing the DOM trees of all its pages to detect fixed patterns. LANTERN by contrast is robust to changes in DOM structure across websites so long as the contextual relationship among attribute nodes don't change much and models trained in one domain can be generalized to other unseen domains.

NLP-based methods, e.g. those utilizing sequence labeling, have also been applied to the attribute extraction task [23, 34]. These techniques are limited to the information present in the textual content of the page however. LANTERN takes advantage of the information in the DOM node types and semantics as well as the structured contextual information present in the DOM tree itself.

Other work [5, 14, 26, 28] has explored leveraging **visual features** of text nodes such as bounding box coordinates and the visual distances to surrounding nodes on the web page. However, obtaining 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 or incompatible with the prevailing browsers. LANTERN builds contextual information directly from the DOM tree without the need for expensive page rendering.

A recent approach called FreeDOM [21] avoids rendering-based features and uses a two-stage model: a first stage that treats the problem as a node-tagging task and a second stage that uses pairwise node relationships to post-process the output from the first stage. In contrast, we propose a simpler, single-stage approach, relying on a careful construction of the context of a node in the DOM

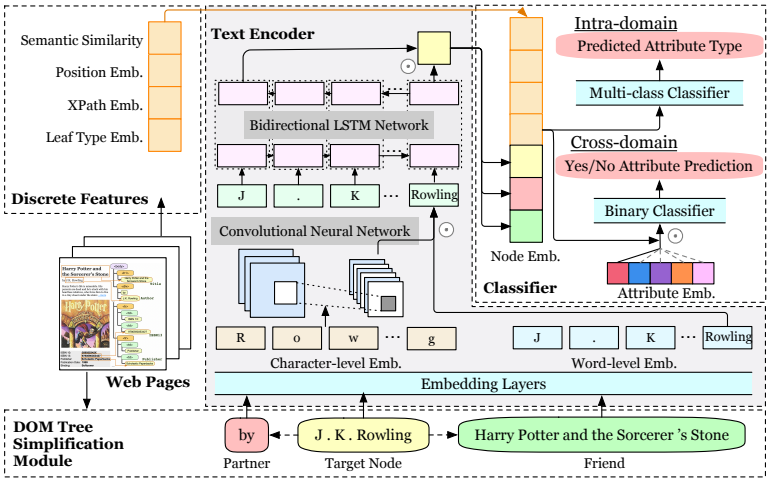

Figure 2: The overall architecture of LANTERN. Nodes' textual features are encoded by LSTM and CNN at the word-level and character-level respectively. A set of discrete features are built from the DOM trees including leaf type, XPath, and the relative position of each node. $[\cdot \odot \cdot]$ denotes vector concatenation.

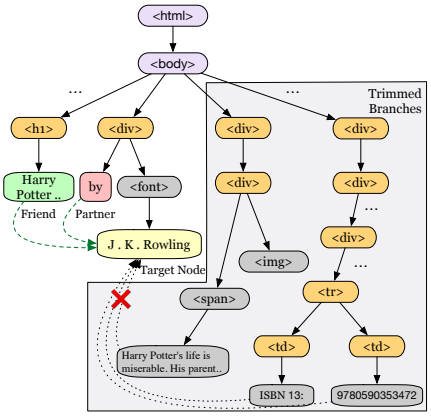

Figure 3: We extract the partner (by) and friends (Harry Potter and the Sorcerer's Stone) for each node (J. K. Rowling) by trimming unrelated branches.

tree. Our approach not only improves the performance of FreeDOM but also generalizes well to unseen websites.

3 PROBLEM FORMULATION AND APPROACH

3.1 Few-shot Attribute Extraction from Semi-structured Websites

We tackle the problem of extracting structured objects from unseen websites. Each *domain* V has a set of websites. Each *website* W is composed of a collection of *detailed pages* which share a similar template. This is a fairly typical assumption since most such web pages are built by instantiating an HTML template with item details that are actually stored in an underlying database.

Attribute Extraction. Given a set of attributes of interest for the target domain, the task at hand is to extract a value (when present) for each attribute from each web page. We make the simplifying assumption that one node can correspond to at most one pre-defined attribute type, consistent with prior work [14]. 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 given attributes (e.g. *title, author, isbn13, publisher*) or *none* representing that this node does not contain any attribute values.

Few-shot Intra-domain Extraction. Given a set of annotated seed websites $\{W_1^a, W_2^a, ..., W_i^a\}$ from domain V, we aim to learn a 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 domain.

Few-shot Cross-domain Extraction. Given a set of annotated seed websites $\{W_1^a, W_2^a, ..., W_i^a\}$ from domain V_1, we aim to learn a 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 domain. However, in this setting, we also have access to annotated websites $\{W_1^{a'}, W_2^{a'}, ..., W_k^{a'}\}$ from domain V_2, where $V2 \neq V_1$.

3.2 Approach Overview

Figure 2 shows the overall framework of the proposed LANTERN model for the few-shot attribute extraction task. First, we extract context features for each node called *friend* and *partner* nodes. Textual features corresponding to each node, its friends, and partners are then fed into a text encoder to generate a dense semantic embedding. We augment this with discrete features built from markup information such as XPath and leaf node types. We then 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 a node. In the intra-domain scenario, we directly apply a multi-class classifier to the node embedding and output the attribute type probability distribution. In the cross-domain scenario, the attribute sets differ from domain to domain. Therefore, we have to alter the inference strategy to binary classification to achieve a matching probability for each attribute type. We select the attribute with the highest probability as the prediction.

Each page has a DOM tree T which contains a variable node set X and a fixed node set Y, where text contents are stored, and also a set of non-text nodes. Fixed nodes remain the same across different detailed pages on the same website (boilerplate like the site's name, navigation elements, etc.) while variable nodes may contain content specific to the object being described on the page. Without loss of generality, we enforce the constraint that the fixed nodes are always mapped to *none*.

4 NODE ENCODER AND CLASSIFIER

The node encoder consists of three components: a module to extract friend and partner nodes from the DOM tree, the text encoder, and a discrete feature module.

4.1 Friend and Partner Nodes

Given a node x in the DOM tree, we define two kinds of nodes *partner* and *friends* that constitute the "friend circle" for the node.

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;
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

The whole DOM tree is a collection of nodes that originate from a unique starting node called the *root*. Recall that 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 such that for each $x^f \in X^F$, the distances from both x^f and x to their lowest common ancestor $a \in A$ is no more than a 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 originate 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. The intuition behind defining partner and friend nodes is simple – while real-world DOM trees can be extremely complicated, most of the *context* for a node is present in DOM nodes that are either friends, and when there's a partner, it contains particularly important context. In Figure 4, we plot a common subtree 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.

Function $\mathcal{F}$, to extract the friend circle features, is 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. Conversely, we can easily obtain all the descendants of each ancestor node to construct 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 3.

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> . 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 4.2 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.

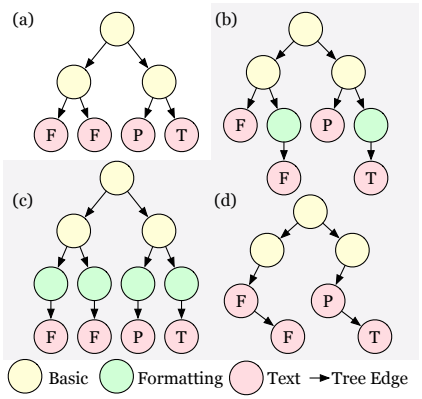

Figure 4: Subtree skeletons of web page DOMs including a common structure (a) and its three possible variants (b), (c) and (d). "Basic" denotes a set of basic HTML element tags, while "Formatting" represents some formatting and style tags such as and <font>. The "Text" node has text information. We aim 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).

4.2 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* domain. Their character-level representation can be very important. Therefore, we leverage a Convolutional Neural Network to encode the character-level embeddings (dimension d_c) of each word w, resulting in h_w^c. We simply concatenate h_w^c with its word-level representation g_w retrieved from external pretrained word embeddings: $h_w = \left[g_w; h_w^c\right]$.

The LSTM [15] has been widely used as the unit of Recurrent Neural Network for learning the latent representation of sequence data [24]. 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 $e_x = \left[h_w^{forward}; h_w^{backward}\right]$.

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

4.3 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* domain, 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 that 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*[1] 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 $d_{xpath}+d_{leaf}+d_{pos}+M$.

4.4 Inference and Optimization

We design different inference strategies for the two scenarios. Under the intra-domain 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-domain scenario, we notice that the MLP layer for multi-class classification can no longer be used for different domains which have different sizes of attribute sets. Therefore, we alter the inference strategy to binary classification. Specifically for each attribute type, we concatenate the node embedding e_n to a randomly initialized attribute embedding e_{a_i} of dimension d_a. We then feed it to a separate MLP and compute a score $\mathbf{h}_i$ to denote the probability of this attribute type:

$$e_{b_i} = \left[e_n; e_{a_i}\right], 1 \le i \le M+1$$

$$\mathbf{h_i} = \text{MLP}(e_{b_i}), \mathbf{h}_i \in \mathbb{R}$$

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

Domain	#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

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 $\mathbf{p}$.

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

5 EXPERIMENTS

In this section, we first 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-domain 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 is the performance of different word embedding strategies?* Lastly, we evaluate the effectiveness of the out-of-domain knowledge under the cross-domain few-shot extraction scenario.

5.1 Dataset

We rely on a public data set, SWDE [14] that consists of more than 124,000 web pages from 80 websites of 8 domains to train and evaluate the proposed model. Detailed statistics are shown in Table 1. Each domain consists of 10 websites and contains 3 to 5 attributes of interest. We notice that the *book* and *job* domains have the most variable nodes on average, roughly three times the number of variable nodes in the *auto* and *university* domains. While SWDE has been around for a while, it has been used in many recent papers [14, 21, 25] and is still a useful dataset to understand the relative merits of different approaches.

In the intra-domain few-shot experiments, we follow the methodology in FreeDOM [21] to select k seed websites as the training data and use the remaining $10-k$ websites as the test set. For example, when $k=2$, we build 10 training sets by picking 10 permutations from all the 2-seed-website combinations such as (*auto*, *book*), (*book*, *camera*), ..., and (*university*, *auto*). The corresponding test set for the first training set is the remaining 8 websites from *camera* to *university*. 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.

In the cross-domain few-shot experiments, we leverage one domain as the out-of-domain knowledge to train a model. Then we conduct the same intra-domain 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 domains, which can tackle the scenario where the domain of the existing annotation is inconsistent with the unseen websites.

5.2 Evaluation Metrics

We evaluate the extraction performance by page-level F1 scores, following the evaluation metrics from SWDE and FreeDOM [14, 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 domains (Table 2) to compare with the baselines. We also compute the average F1 score for each domain (Figure 5) and each attribute (Figure 6) for detailed analysis.

5.3 Implementation details

For data pre-processing, we use the open-source LXML library[2] to extract DOM tree structures from each page. Then, we follow the simple heuristic 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 training speed is significantly accelerated. We use GloVe pretrained representations [32] to initialize our word embeddings. Other representations such as character embeddings and attribute embeddings are all randomly initialized. We also truncate every node's text to a maximum of 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 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 a kernel size of 3. For the LSTM network, we set the hidden layer size as 100. The model is implemented in Tensorflow[3]. We train the model using 15 epochs and a batch size of 32. We apply a dropout mechanism following the MLP layer to avoid over-fitting issues. The dropout rate is 0.3. We use Adam [17] as the optimizer with a learning rate of 0.001. It takes less than 30 minutes to finish the complete training and evaluation cycle for each domain with one NVIDIA V100 GPU.

5.4 Baseline Models

We compare against several baselines:

Stacked Skews Model (SSM). SSM [5] utilizes expensive handcrafted features and tree alignment algorithms to align the unseen web pages with seed web pages. Attribute values are extracted from each page of the unseen websites. Like our model, this method does not require visual rendering features.

[2]https://lxml.de/

[3]https://www.tensorflow.org/

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
LANTERN	83.06	**88.96**	**91.63**	**92.84**	**93.75**

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

Rendering-feature Model (Render-full). Render-full [14] employs visual features to express the distances between node blocks rendered with the web browser. Visual distances have proven to be a good method to encode the neighboring relationships among nodes [26] but this method requires the time-consuming rendering process and needs extra memory space to save the images, CSS, and JavaScript that can easily be out-of-date. Render-full employs a sophisticated heuristic algorithm to compute the visual distances, which gives the best performance [14], 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 deploy in practice. Additionally, 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.

5.5 Intra-domain Few-shot Extraction Results

Table 2 shows the overall comparisons between our model LANTERN and all four baselines using different numbers of seed websites. Our model achieves slightly worse performance when $k = 1$ while it 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 improvement of 1.44% over all values of k) and achieves a 3.47%-10.54% improvement over the single-stage approach, FreeDOM-NL.

We plot the detailed performance of LANTERN on different domains in figure 7. In general, the performance improves as k increases. This is not surprising because more training data yields better coverage of all possible instances. We also observe that the rate of performance growth slows down and sometimes the F1

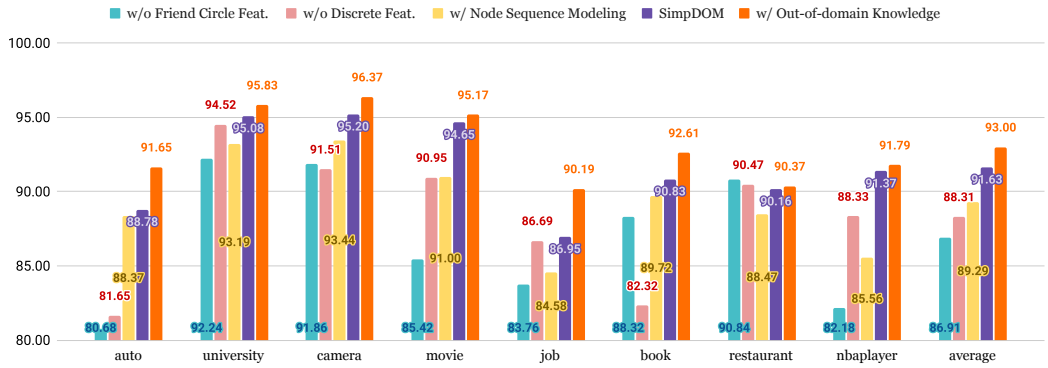

Figure 5: Ablation study results that demonstrate the contribution from different features and modules. We conclude that both friend circle features and discrete features improve the extraction performance while adding a sequence modeling module harms the performance dramatically. With the out-of-domain knowledge from a second domain, the model can do better for each of them. We set $k = 3$ here. Similar results can be achieved with other k's.

Figure 6: Per-attribute F1 performance comparisons between LANTERN 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 improvements.

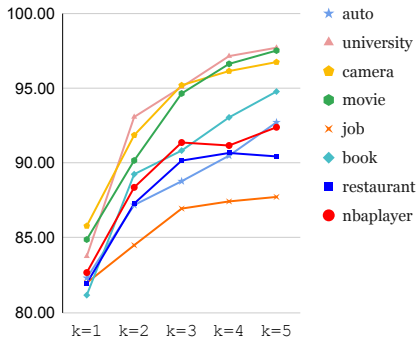

Figure 7: Comparing the extraction performance (F1 score) of different numbers of seed sites $k = \{1, 2, 3, 4, 5\}$ per domain.

scores of some domains (e.g. *nbaplayer* and *restaurant*) even fluctuate as more data is added to the training set (i.e. as k increases). We surmise that the reason for this behavior is that the model becomes more robust and less new knowledge can be transferred from annotated websites to unseen websites in these domains.

5.6 Ablation Study

In Figure 5, we present an ablation study on different features of LANTERN, 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 increase the F1 score of the *nbaplayer* domain from 82.18% to 91.37% and the discrete features increase the performance on the *book* domain by 8.51%. However, *restaurant* is a special case where the score 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 adds more noise to the classification. This is also corroborated by Figure 6, which explains the detailed performance change when adding the friend circle features per attribute. We observe that the

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 $k = 3$.

improvement on *height* of *nbaplayer* is significant. The nodes containing *height* value always share a similar pattern xx-yy[4] 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. The extraction of some attributes such as *company* in the *job* domain and *address* in the *restaurant* domain was not improved. We believe this is caused by the comparatively diverse positions of these attributes in different websites.

Another interesting ablation study is done with an additional sequence modeling layer[5] which is commonly applied to sequence labeling tasks such as named entity recognition on plain text [20, 43]. 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 5 (marked as "w/ Node Sequence Modeling"), the additional sequence modeling layer fails to optimize the node representations for all the domains especially those with more variable nodes such as *nbaplayer* and *job*. We believe that the information from all other DOM tree nodes can be selectively attended to the current node with this mechanism, however this introduces more noise than useful knowledge. This further demonstrates the importance of utilizing the structure of the DOM tree to eliminate noise from distant and irrelevant nodes.

We also compare 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, initialized from GloVe, achieving the best performance. We think that a specific "web-language" model can serve the web information extraction tasks better. Drawing on recent developments in contextualized language models, we also tried using BERT [9][6] to generate the contextualized embeddings but it decreases the performance by 9.8%. This is not surprising given that the context in each node is very limited[7] and the huge size of parameters (110M in BERT-BASE) for fine-tuning can easily cause an over-fitting problem.

5.7 Cross-domain Few-shot Extraction Results

We plot a heatmap in Figure 8 to denote the performance improvements from the out-of-domain knowledge. Each entry in the heatmap relates to a pair of domains, where the domain in the upper case is used as the out-of-domain knowledge while the domain in the lower case is used to train and test the model. We do not plot the scores in the diagonal because domains cannot serve as their own 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 domains. 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 domain achieved by using the most helpful out-of-domain knowledge in Figure 5. We achieve the highest average F1 score of 93% over all the domains ($k = 3$), which improves the performance of our intra-domain experiment by a further 1.37% (absolute value). This evidence proves our assumption that a better contextual node representation can be learned from additional knowledge, which is extremely helpful when only a few labeled data are provided for specific domains.

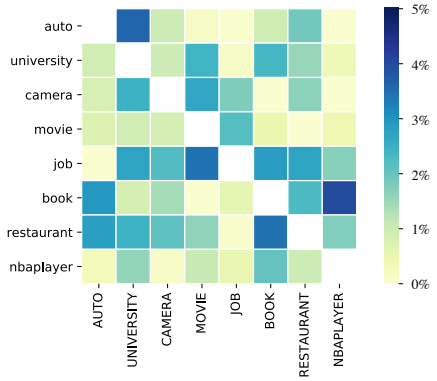

Figure 8: Heatmap denoting the performance improvements per F1 score from the out-of-domain knowledge ($k = 3$). We learn a transferable model with domains in upper case (columns). Then we fine tune the model and predict on the domains in lower case (rows). We observe that over 20 pairs of websites improve the extraction performance by 1.5%.

6 CONCLUSION AND DISCUSSION

In this paper, we present a simple but effective method, LANTERN, for the attribute extraction task. LANTERN uses the tree structure of the neighborhood around a node to learn a contextual representation for each node in the DOM tree. This method does not require the expensive generation of visual features and is more robust than wrapper induction. This opens up interesting directions for future investigation – since we're able to exceed the accuracy of previous SOTA without using visual features, are there other techniques that better exploit visual features? When does the additional cost of processing rendered visual features justify increased accuracy?

Additionally, we show that using training data from a different domain is surprisingly useful and improves performance by a *further* 1.37% relative to the SOTA method. These novel cross-domain, few-shot extraction results demonstrate the potential of our approach to generalize across domains and the ability to use data from an existing domain to rapidly bootstrap an extraction model for new domains.

[4]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.
[5]We utilize Transformer [40] as the sequence modeling layer. LSTM is another option.
[6]We choose BERT without loss of generality. We can also use ELMo or XLNet [33, 44].
[7]On average, each variable node contains only 2-5 words in different domains.

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] 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.

[4] M. Bronzi, V. Crescenzi, P. Merialdo, and P. Papotti. Extraction and integration of partially overlapping web sources. *Proceedings of the VLDB Endowment*, 6(10):805–816, 2013.

[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] 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.

[8] V. Crescenzi and P. Merialdo. Wrapper inference for ambiguous web pages. *Applied Artificial Intelligence*, 22(1-2):21–52, 2008.

[9] 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.

[10] 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.

[11] 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.

[12] T. Furche, G. Gottlob, G. Grasso, X. Guo, G. Orsi, C. Schallhart, and C. Wang. Diadem: thousands of websites to a single database. *Proceedings of the VLDB Endowment*, 7(14):1845–1856, 2014.

[13] 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.

[14] 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.

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

[16] M. Kayed and C.-H. Chang. Fivatech: Page-level web data extraction from template pages. *IEEE transactions on knowledge and data engineering*, 22(2):249–263, 2009.

[17] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[18] 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.

[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. Ã. Ling Liu. *Encyclopedia of Database Systems*. Springer New York, 2nd ed. edition, 2018.

[23] H. Liu, C. Chen, L. Zhang, and G. Wang. The research of label-mapping-based entity attribute extraction. In *2010 IEEE International Conference on Progress in Informatics and Computing*, volume 1, pages 635–639. IEEE, 2010.

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

[25] 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.

[26] 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.

[27] 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.

[28] A. More. Attribute extraction from product titles in ecommerce. *arXiv preprint arXiv:1608.04670*, 2016.

[29] 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.

[30] A. Parameswaran, N. Dalvi, H. Garcia-Molina, and R. Rastogi. Optimal schemes for robust web extraction. *Proceedings of the VLDB Conference*, 4(11), September 2011.

[31] 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.

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

[33] 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.

[34] D. Putthividhya and J. Hu. Bootstrapped named entity recognition for product attribute extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1557–1567, 2011.

[35] 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.

[36] H. A. Sleiman and R. Corchuelo. Trinity: on using trinary trees for unsupervised web data extraction. *IEEE Transactions on Knowledge and Data Engineering*, 26(6):1544–1556, 2013.

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

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] 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.

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

[44] 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.

[45] 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.