Self-Training for Label-Efficient Information Extraction from Semi-Structured Web-Pages

Ritesh Sarkhel
Amazon
Seattle, Washington

Binxuan Huang
Amazon
Seattle, Washington

Colin Lockard
Amazon
Seattle, Washington

Prashant Shiralkar
Amazon
Seattle, Washington

ABSTRACT

Information Extraction (IE) from semi-structured web-pages is a long studied problem. Training a model for this extraction task requires a large number of human-labeled samples. Prior works have proposed transferable models to improve the label-efficiency of this training process. Extraction performance of transferable models however, depends on the size of their fine-tuning corpus. This holds true for large language models (LLM) such as GPT-3 as well. Generalist models like LLMs need to be fine-tuned on in-domain, human-labeled samples for competitive performance on this extraction task. Constructing a large-scale fine-tuning corpus with human-labeled samples, however, requires significant effort. In this paper, we develop a ***La**bel-**E**fficient **S**elf-**T**raining Algorithm* (LEAST) to improve the label-efficiency of this fine-tuning process. Our contributions are two-fold. *First*, we develop a generative model that facilitates the construction of a large-scale fine-tuning corpus with minimal human-effort. *Second*, to ensure that the extraction performance does not suffer due to noisy training samples in our fine-tuning corpus, we develop an uncertainty-aware training strategy. Experiments on two publicly available datasets show that LEAST generalizes to multiple verticals and backbone models. Using LEAST, we can train models with less than ten human-labeled pages from each website, outperforming strong baselines while reducing the number of human-labeled training samples needed for comparable performance by up to $11x$.

PVLDB Reference Format:
Ritesh Sarkhel, Binxuan Huang, Colin Lockard, and Prashant Shiralkar. Self-Training for Label-Efficient Information Extraction from Semi-Structured Web-Pages. PVLDB, 16(11): XXX-XXX, 2023.
doi:XX.XX/XXX.XX

1 INTRODUCTION

Semi-structured web-pages are great sources of high-quality information. Extracting structured information from them is a well-studied problem with a broad range of applications, including recommendation systems [49], product search [5], knowledge-base construction [8, 15], question answering [12], and more. We focus on an information extraction (IE) task from *semi-structured detail-pages* in this paper. A detail-page is a web-page that represents a single entity, which is typically the topic entity of that page. Take the detail-page in Fig. 1 for example. We want to extract text-spans corresponding to each attribute in a predefined attribute-set from this page. The semi-structured format and diverse layout of these pages however, make this task challenging to scale for a large-scale corpus.

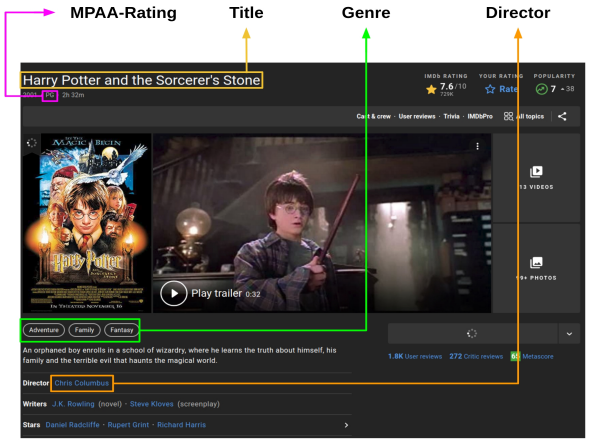


Figure 1: A web-extraction model extracts a set of attributes from a semi-structured web-page. In this example, we extract the attributes {*title, director, genre, mpaa-rating*} from a detail-page belonging to the `movie` vertical. For each attribute, we identify a set of text-spans appearing on that page.

Example: Consider the following example. *Alice* wants to construct an authoritative knowledge-base for answering questions about products sold on various e-commerce websites. She has a list of some of the most popular e-commerce websites. New websites get added to this list periodically. Each website on this list comprises of detail-pages with diverse layouts covering multiple domains (e.g. books, electronics). Layout of each page varies between websites and/or domains. To construct a knowledge-base from this diverse set of websites, we need to extract a set of attributes from the detail-pages of each website. The information extraction (IE) task that needs to be undertaken for this purpose needs to be both scalable and robust. Traditional IE methods such as wrapper induction [20, 26] is hard to scale for this task. To understand why,

Proceedings of the VLDB Endowment, Vol. 16, No. 11 ISSN 2150-8097.
doi:XX.XX/XXX.XX

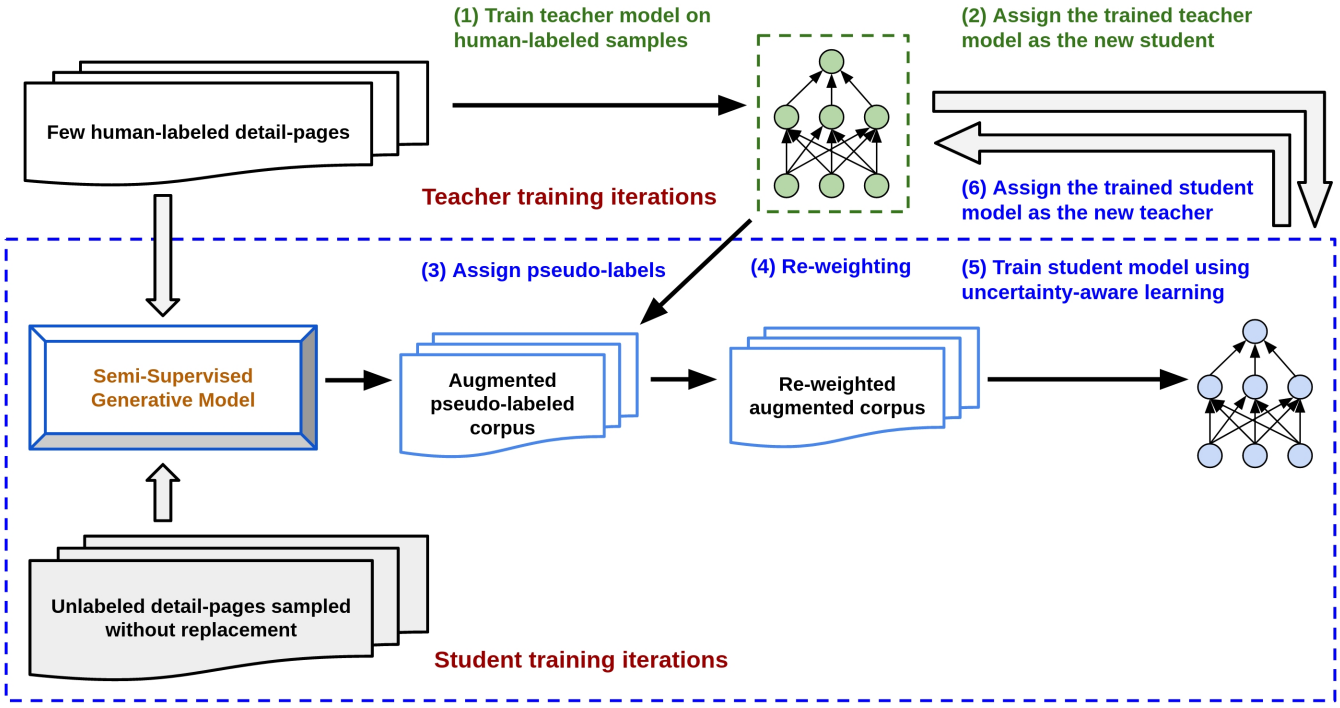


Figure 2: At each iteration, LEAST trains a teacher model on a few human-labeled samples in the corpus. Once training terminates, the teacher model along with a semi-supervised generative model pseudo-annotates some unlabeled samples from the same corpus. The student model is initialized with the weights of the teacher model. LEAST trains the student model on an augmented corpus containing both human-labeled and pseudo-labeled samples using an uncertainty-aware training strategy. Once training terminates, the student model initializes the weights of the teacher model for the next iteration. This process is repeated until both models converge.

let's take a look at how it works. Wrapper induction methods identify text-spans corresponding to an attribute by deriving a set of XPath-based rules or wrappers from human-labeled samples. Unfortunately, these wrappers do not generalize for web-pages that have different layouts than the human-labeled samples used to infer them [30]. This leads to degraded extraction performance and triggers the need of additional human-labeling for increased coverage. As more websites with diverse layouts are added to the list, this overhead compounds quickly for customers like *Alice*. We propose an end-to-end training algorithm that addresses this challenge by minimizing the number of human-labeled pages needed to train a generalizable web-extraction model without sacrificing its performance.

Motivation. Prior works have explored different avenues to improve the label-efficiency of web-extraction models. Some researchers [19, 30] have leveraged distance supervision to label web-pages using an external knowledge-base. A comprehensive knowledge-base however may not always be available, especially for emerging domains. An extraction model trained on distant labels may not generalize for attributes at the long-tail of the distribution as well. Recent works have tried to address this by developing transferable models [21, 29, 54], where they pretrain a model on thousands of human-labeled pages from some seed websites first, and then fine-tune [37] it on human-labeled pages from the target websites. Factors like multilingual content, low page-level consistency, and lack of inter-annotator agreement make labeling web-pages with diverse layouts a resource-intensive process. Pretraining on unlabeled pages using self-supervision [13, 27, 36] improves the label-efficiency of the pretraining process to some extent. Extraction performance of a model trained using self-supervision, however, still relies on the number of human-labeled pages in the fine-tuning corpus. The same is true for large language models (e.g. GPT-3 [7]). Although they generalize well for a wide range of natural language related tasks in few-shot settings [7], for visually rich complex documents [41–44] such as semi-structured detail-pages, they need to be fine-tuned on in-domain human-labeled samples [1, 16] for competitive extraction performance. Improving the label-efficiency of a robust, transferable web-extraction model, therefore requires addressing two key questions.

(i) how to minimize the number of human-labeled pages (from both seed and target websites) during training?
(ii) how to achieve (i) without affecting the model's extraction performance?

Challenges. Recent works [24, 28] have shown the efficacy of self-trained models for classification tasks in limited labeled data scenarios. The traditional self-training algorithm [45] comprises of two models – a teacher model and a student model. At each iteration, the teacher model is trained on some human-labeled samples (H) in the training corpus first. Once trained, it annotates some

unlabeled samples (U) from the same corpus and initializes the weights of the student model. The student model is then trained on a corpus containing both human-labeled and pseudo-labeled samples ($H \cup U$) inferred by the teacher model. Once trained, the student model initializes the weights of the teacher model for the next iteration. These steps are repeated until both models converge. Unfortunately, one of the major drawbacks of this algorithm, especially when the number of human-labeled samples is limited, is ensuring the quality of the final model. This is because pseudo-labeled samples inferred by the teacher model that has been trained on a small human-labeled corpus can be quite noisy. Training the student model on such noisy samples results in gradual drift [40] in the model's quality due to error propagation during the gradient update step (discussed in Section 2). We develop an uncertainty-aware training strategy to address this challenge in this paper.

Contributions. Our main contributions are two-fold. *First*, we ensure that the quality of pseudo-labeled samples in our training corpus is high by developing a semi-supervised generative model. It acts as a supplementary supervision source (Section 4.1) along with teacher model to infer high-quality pseudo-labels. *Second*, we ensure that our model does not degrade in quality due to noisy, pseudo-labeled samples in our corpus by developing an *uncertainty-aware training strategy*. We estimate the label-uncertainty of each sample in our corpus and use it to re-weight gradient updates at each training iteration (Section 4.2). We assign higher (or lower) weights to amplify (or dampen) training signals from less (or more) noisy samples. Contrary to prior works [24, 34], we do not solely rely on prior knowledge about the content of a training sample to infer its weight. This makes our method robust towards out-of-distribution samples in our training corpus. Once training terminates, the final model undertakes a multi-class classification task to categorize each DOM-node in the test corpus as one of the attributes to be extracted (or `None`). We combine all of the components described above within a single framework, called `LEAST`. Figure 2 presents an overview of its end-to-end workflow.

Summary of results. `LEAST` is an end-to-end training algorithm that is compatible with multiple contemporary web-extraction models. We exhibit this capability by training multiple public models using `LEAST`. We evaluate their performance on two publicly available datasets – the Structured Web Data Extraction (SWDE) dataset [22], and the Web-based Structural Reading Comprehension (WebSRC) dataset [10]. Our results show that `LEAST` generalizes to multiple datasets and backbone models. Using `LEAST`, we can train models with less than ten human-labeled pages from each seed website and outperform state-of-the-art baselines by up to 22 F1 points – reducing the number of human-labeled samples needed to train the same model for comparable extraction performance using a naive transfer learning baseline [29, 54] by up to $11x$.

2 BACKGROUND

Before presenting our workflow, we introduce some of the concepts and formalizations used to describe it first.

A. **Websites:** A website is a collection of web-pages that share similar HTML templates. Each website used in our experiments is either a *seed* or a *target* website. Typically, a seed website is a popular/well-known website in a particular domain, whereas a target website can be a new/emerging website. Due to this qualitative difference, the cost of human-labeling also differs from one website to another. There is an abundance of preexisting resources that can be leveraged to acquire human-labeled pages from some of the most popular websites in a particular domain. Unfortunately, this is not the case for emerging websites. Therefore, additional resources are needed to set up a human-in-the-loop workflow for labelling such websites from the scratch.

B. **Detail-Pages:** Each website comprises of multiple detail-pages. A detail-page is a web-page that represents a single entity, which is typically the topic entity of that page. The topics covered by detail-pages in each website have one-to-one mapping to a real-world object referred as the *target vertical*. For example, the detail page shown in Fig. 1 belongs to a website from the `movie` vertical.

C. **IE from Detail-Pages:** Our objective is to extract a set of attributes A' from detail-pages appearing in a target website. The set of attributes to be extracted is specific to each vertical. Following prior works [22, 29], we formulate this task as a multi-class classification problem. Extracting A' from a detail-page, therefore boils down to inferring a softmax label for each DOM-node on the page, between the attributes defined in A' and `None`. We assume that each DOM-node in a target website corresponds to at most one attribute.

D. **Self-Training Web-Extraction Models:** The traditional self-training algorithm [45] consists of a teacher model $f(.;\theta_{tea})$ and a student model $f(.;\theta_{stu})$. At each iteration t, the teacher model is trained on a randomly sampled human-labeled corpus H. The number of samples in H is typically kept constant for each iteration [3]. Once training terminates, the teacher model annotates a randomly sampled unlabeled corpus (U) and initializes the weights of the student model, (i.e. $\theta_{stu}^{(t-1)} = \theta_{tea}^{(t)}$). The student model is then trained on the augmented corpus $H \cup U$ containing both human-labeled and pseudo-labeled samples inferred by the teacher model. Both models update their weights using backpropagation (see Eq. 1) in each iteration. Once training terminates, the student model initializes the weights of the teacher model to start the next iteration. We train both models in this alternating fashion until convergence.

$$\theta^{(t)} = \theta^{(t-1)} - \alpha \cdot \nabla \frac{1}{N} \Sigma_{m=1}^{N} \mathcal{L}(y_m, f(x_m, \theta^{(t-1)})) \quad (1)$$

In Eq. 1, $\mathcal{L}(,)$ is a cross-entropy-based loss function, y_m denotes the groundtruth (human-labeled or pseudo-labeled) for the training sample x_m. α denotes the learning-rate and $\nabla(.)$ denotes the gradient operator. $\theta^{(t)}$ represents model weights of the teacher (or the student) model at iteration t. Each training sample is represented as a nested tuple $\{x_m, y_m, p_m, w_m\}, \forall m \in [1, N]$, where x_m represents a DOM-node, y_m represents its groundtruth label, p_m denotes the detail-page, and w_m denotes the website on which it appears.

E. **Transferable Web-Extraction:** Training a model using direct supervision on a large-scale corpus with diverse page layouts can require thousands of human-labeled pages from a target

website for robust extraction performance. Annotating webpages with diverse layouts is a resource-intensive task. Recent works [27, 54] have leveraged transfer learning [37] to improve the label-efficiency of this training process. They mitigate labeling cost by pretraining the model on training samples from some seed websites first, and then fine-tune it on a few human-labeled samples from each target website, thus transferring the knowledge gleaned from seed websites to target websites with the help of a few human-labeled samples.

3 PROBLEM DEFINITION

Given n semi-structured websites W, our objective is to train a web-extraction model $\mathcal{E}$ that extracts a set of attributes A' from a set of target websites $W_{target} \subset W$ using minimal number of human-labeled samples from W to train $\mathcal{E}$ without sacrificing its downstream performance.

We propose LEAST – a self-training algorithm that optimizes these two seemingly contradictory objectives simultaneously by minimizing a cross-entropy based loss function on a pseudo-labeled corpus. We describe each component of our self-training algorithm and its workflow next.

4 METHODOLOGY

One of the shortcomings of the self-training algorithm is ensuring model quality when the number of human-labeled samples is small. Pseudo-labeled samples inferred by a teacher model that has been trained on a small human-labeled corpus is often quite noisy [40]. A student model trained on these samples degrades in quality due to error propagation during the gradient update steps (Eq. 1). We address this shortcoming by following a two-prong approach. *First*, we develop a *semi-supervised generative model* that acts as a supplementary supervision source and infers high-quality pseudo-labeled samples with minimal human supervision. *Second*, we develop an *uncertainty-aware training strategy* that re-weights each sample based on its label-uncertainty and amplifies (or dampens) training signals obtained from it. We describe both of them below.

4.1 Supplementary Supervision Source

Following prior works [6], we assume that each website w_i publishes detail-pages by applying a site-specific HTML template to a noisy dynamic view [25] $\mathbb{H}_{w_i}$ of a vertical-specific abstract relation $\mathbb{H}$. In other words, $\mathbb{H}$ is a materialized relation with abstract data types [35] as columns. In this construct, information extraction from a detail-page boils down to inverting this generative process to infer the website-specific relation $\mathbb{H}_{w_i}$. Assuming partial overlap between websites within a vertical, these website-specific relations can then be used as a distance supervision source to label detail-pages from websites in the same vertical.

4.1.1 Formalization. We assume that a website $\mathbf{w}_i$ publishes detail-pages from a vertical-specific abstract relation $\mathbb{H}$ as follows.

$$\mathbf{w}_i = R_i(e_i(\pi_i(\sigma_i(\mathbb{H})))) \tag{2}$$

In Eq. 2, σ_i represents the `selection` operator – it returns a subset of tuples in $\mathbb{H}$, π_i represents the `projection` operator – it returns a subset of columns in $\mathbb{H}$, e_i represents website-specific page-level noise – it returns a relation with extraneous and/or erroneous attribute-values, R_i represents a website-specific rendering function – it encodes each tuple of the website-specific relation $\mathbb{H}_{w_i} = e_i(\pi_i(\sigma_i(\mathbb{H})))$ as a detail-page in website $\mathbf{w}_i$.

4.1.2 Inverse rendering rules. The objective of our generative model γ is to infer a website-specific relation $\mathbb{H}_{w_i}$ from a few human-labeled pages of a website $w_i, \forall i$. We formulate this task as inferring a set of *website-specific inverse rendering rules*. Formally, an inverse rendering rule is a one-to-one mapping between an attribute and a set of DOM-nodes in w_i. We follow a two-pronged approach to infer these rules. They are as follows.

A. *Page-level consistency.* We encode page-level consistencies to capture inverse rendering rules in the form of weakly supervised functions [43, 52]. Each function takes as input an attribute from the vertical-specific attribute-set A', an unlabeled detail-page p, and a list of human-labeled DOM-nodes (*node_list*) that are annotated as true instances of that attribute. It outputs a list of DOM-nodes from an unlabeled page p corresponding to that attribute. These functions can range from open-source libraries [33], regular expressions encoding textual/XPath-based patterns to custom functions that combine multiple modalities. The number and type of functions employed depend on the website and the target vertical. We provide an example below.

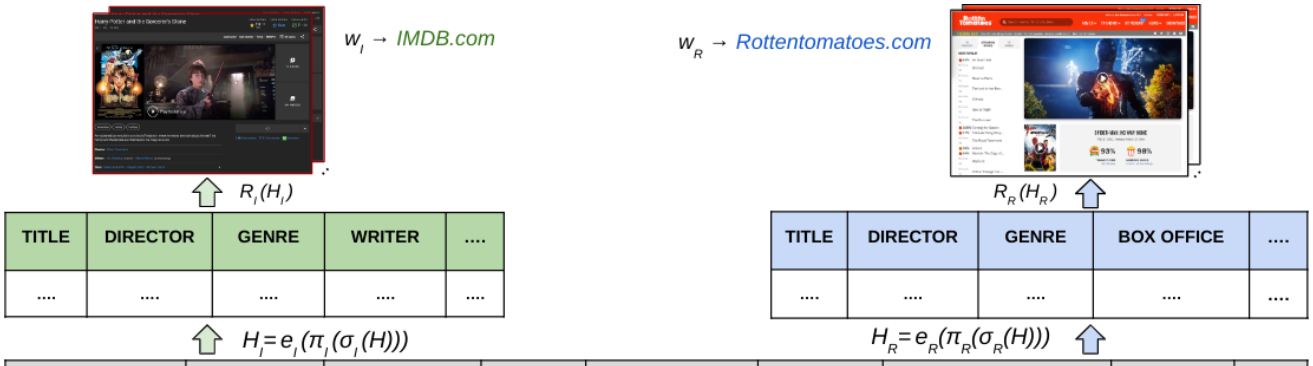
```
# Fuzzy string matching
        def fuzzy_string_matcher(node_list,a,p){
            labeled_node_list = []
            for node in p.dom_tree():
                if node.hasText():
                    for n in node_list:
                        if fuzzy_match(n.text(),node.text()):
                            labeled_node_list.append(node)
            return labeled_node_list
        }
```

In this example, we use fuzzy string matching to assign each DOM-node of an unlabeled page with labels from the vertical-specific attribute-set.

B. *Overlapping attributes. Web-Extraction and Integration of Redundant data* or `WEIR` [6] is a publicly available library[1] that infers website-specific relations ($\mathbb{H}_{w_i}$) by aligning partially overlapping detail-pages across different websites in the vertical. By identifying overlapping text elements in various websites, it aligns detail-pages representing similar topics and infers a set of rules for those entities. We bootstrap these rules using a few human-labeled pages from each website w_i to infer inverse rendering rules for each attribute in A'.

4.1.3 Inferring pseudo-labels. We employ the inverse rendering rules learned this way to infer website-specific relations $\mathbb{H}_{w_i}, \forall i$. We use these relations as a distance supervision source to annotate unlabeled pages in our corpus. If a DOM-node with text t is an instance of an attribute $a \in A'$ in a website-specific relation $\mathbb{H}_{w_i}$, we label all DOM-nodes in an unlabeled page with text T as an instance of the attribute a. In case of different labels are assigned to T in different website-specific relations, we assign a label by majority voting.

[1] https://github.com/crescenz/weir



H

TITLE	DIRECTOR	GENRE	MPAA RATING	WRITER	RELEASE DATE	ACTOR	BOX OFFICE	....
Harry Potter and the Sorcerer's Stone	Chris Columbus	Adventure, Family, Fantasy	PG	J.K. Rowling, Steve Kloves	2001	Daniel Radcliffe, Rupert Grint, Richard Harris	$317.9M	....
Jurassic Park	Steven Spielberg	Action, Adventure, Sci-Fi	PG-13	Michael Crichton, David Koepp	1993	Sam Neill, Laura Dern, Jeff Goldblum	$401.8M	....
Spider-Man: No Way Home	Jon Watts	Action, Adventure, Fantasy	PG-13	Chris Mckenna, Erik Sommers, Stan Lee	2021	Tom Holland, Zendaya, Benedict Cumberbatch	$702.9M	....
....	....	....	....	....	....	....	....	....

Figure 3: The generative model renders a semi-structured website from a noisy dynamic view of a relation H using website-specific HTML templates. We infer inverse rendering rules to obtain website-specific relations H_I and H_R that act as a distant supervision source in our training workflow.

4.2 Uncertainty-Aware Training

Training a model on noisy pseudo-labeled samples can result in gradual drifts [40] in the model's quality. Errors due to label-noise can propagate through the model during gradient updates (see Eq. 1). To mitigate this, we adopt an uncertainty-aware training strategy. It has two main components. They are as follows.

4.2.1 Adaptive re-weighting. Leveraging the idea of weight perturbation [39], we assign each training sample x_m a weight $c_m \in (0, 1]$. We amplify (or dampen) the training signal for a sample x_m by multiplying it with a weight proportional to its estimated label uncertainty. We assign smaller weights to samples that are noisy and larger weights to samples that are less noisy. Training samples appearing on the same detail-page are assigned the same weight. Both the teacher and the student model are trained on this re-weighted corpus, updating their weights after each iteration t as follows.

$$\theta^{(t)} = \theta^{(t-1)} - \alpha \cdot \nabla \frac{1}{N} \Sigma_{m=1}^{N} (c_m \cdot \mathcal{L}_{na}(y_m, f(x_m, \theta^{(t-1)}))) \quad (3)$$

In Eq. 3, $\mathcal{L}_{na}(.)$ denotes a *noise-aware loss function* (defined next). The weight $c_m \in (0, 1]$ assigned to a sample depends on a number of factors. If x_m appears on a human-labeled web-page, we assign it a weight of 1. If it appears on an unlabeled page but the website has some human-labeled pages in our corpus, we assign it a weight equal to the validation accuracy of the student model on that website. Finally, if x_m appears on an website that has no human-labeled pages in our corpus, we assign it a weight $c_m = c_j \cdot JS(p_m, p_j)$, where $j = \texttt{argmax}_i\, JS(p_m, p_i)$ and c_j denotes the weight assigned to the human-labeled page p_j that has the highest Jaccard-Similarity with the web-page on which x_m appears. We represent each web-page as a bag-of-words to compute the Jaccard-Similarity score.

4.2.2 Noise-aware loss. We incorporate the sample weights computed above as an uncertainty measure to compute a noise-aware loss for each training sample. For a sample x_m with a softmax label $\hat{y}_m^{(t)}$ at training iteration t and groundtruth y_m, we compute this loss term as follows.

$$\mathcal{L}_{na}(\hat{y}_m^{(t)}, y_m) = \mathcal{L}(\hat{y}_m^{(t)}, y_m) \cdot exp(1 - c_m \cdot k^{(t)}) + exp(c_m) \cdot U(0, 1) \quad (4)$$

In Eq. 4, $\mathcal{L}(.)$ is a cross-entropy-based loss function, c_m represents the sample-weight for x_m, and $k^{(t)} \geq 1$ represents a penalty term that accounts for distribution shift during training. The second term is a regularization factor to prevent the model from overfitting and $U(0, 1)$ is a random number within $[0, 1]$. Setting $k^{(t)}$ to a larger number increases the penalty imposed on the model and encourages it to learn more from out-of-distribution samples. We update $k^{(t)}$ after each iteration to regulate this as follows.

Algorithm 1 The LEAST Algorithm

Input: Human-labeled samples $H = (X^l, Y^l, P^l, W^l)$; Unlabeled samples $U = (X^u, P^u, W^u)$; Maximum number of pseudo-labeled samples L; Number of training iterations T.
Output: Final model $f(:, \theta_{stu}^{(T)})$
Initialize teacher training corpus $C_{tea}^{(0)} = H$
Initialize student training corpus $C_{stu}^{(0)} = H$
for $i = 1$ to T **do**
  Train the teacher model $f(:, \theta_{tea}^{(t)})$ on $C_{tea}^{(0)}$
  Sample L unlabeled samples from U without replacement
  for each $(x^u, p^u, w^u) \in U$ **do**
    Infer a pseudo-label $\hat{y}^{(t)}$ for x^u
    Update $C_{stu}^{(t)} = C_{stu}^{(t-1)} \cup (x^u, \hat{y}^{(t)}, p^u, w^u)$
  end for
  for each $(x, \hat{y}^{(t)}, p, w) \in C_{stu}^{(t)}$ **do**
    Compute a sample weight for x
  end for
  Initialize the student model using weights of $f(:, \theta_{tea}^{(t)})$
  Train the student model on $C_{stu}^{(t)}$ using noise-aware loss
  Update the teacher model $f(:, \theta_{tea}^{(t+1)}) = f(:, \theta_{stu}^{(t)})$
end for

$$k^{(t)} = k^{(t-1)} - \mathcal{K}_1 \cdot exp(-\mathcal{K}_2 \cdot t) \tag{5}$$

In Eq. 5, $\mathcal{K}_1, \mathcal{K}_2 \geq 0$ are constants. At each training iteration t, we update the weights of the teacher and the student model (Eq. 3) with gradients computed using this loss function. We present a sensitivity analysis of the hyperparameter $k^{(t)}$ on a LEAST-trained model's extraction performance in Appendix B.

4.3 End-to-End Workflow

4.3.1 Training workflow. We provide an overview of our training workflow in Algorithm 1. We start by training the teacher model on a few randomly sampled human-labeled samples. Once training terminates, we initialize the student model with weights of the teacher model. We construct the student training corpus by augmenting a few human-labeled samples (H) with a large number of pseudo-labeled samples (U). We construct the pseudo-labeled corpus by assigning each unlabeled DOM-node in our training corpus a softmax label. At each training iteration t, we combine two supervision sources for this purpose: (a) the teacher model $f(.; \theta_{tea}^{(t-1)})$, and (b) the generative model γ. We assign an unlabeled DOM-node a pseudo-label inferred by the generative model with a probability of $\beta^{(t)}$, or the teacher model with a probability of $(1 - \beta^{(t)})$. We prioritize the pseudo-labels inferred by the generative model during the early iterations and the teacher model during the later iterations of our self-training algorithm. This reduces the likelihood of label noise propagating through the student model during the gradient update step. We update the hyperparameter $0 < \beta^{(t)} < 1$ after each training iteration as follows.

$$\beta^{(t)} = \beta^{(t-1)} - \mathcal{B}_1 \times exp(-\mathcal{B}_2 \cdot t) \tag{6}$$

In Eq. 6, $\mathcal{B}_1, \mathcal{B}_2 \geq 0$ are constants. We present a sensitivity analysis of the hyperparameter $\beta^{(t)}$ on a LEAST-trained model's extraction performance in Appendix B. We present the value of all LEAST hyperparameters used in our experiment in Table 1. We obtain these values empirically through grid search. Once the student model is trained, we initialize the teacher model with weights of the student model for the next iteration. We repeat this process until both models converge. We train both models on a re-weighted corpus using the noise-aware loss function defined in Eq. 4. Combining two supervision sources to infer the pseudo-labeled samples ensures that we have a large-scale high-quality corpus. Re-weighting each training sample by estimating its label uncertainty ensures that error propagation from noisy, pseudo-labeled samples is mitigated during model training. We measure the individual contribution of various components in our training workflow by performing an ablation study in Section 5.4.

4.3.2 Inference workflow. Once training terminates, the final model boils down the IE task to a multi-class classification problem. It assigns each DOM-node with non-empty text in our test corpus a softmax label from the vertical-specific attribute-set A' or None.

Table 1: Values of different hyperparameters in LEAST

Symbol	Description	Value
T	Maximum no. of self-training iterations	5
$\beta^{(0)}$	Pseudo-labeled corpus construction	0.6
$\mathcal{B}_1$	Update $\beta^{(t)}$	0.1
$\mathcal{B}_2$	Update $\beta^{(t)}$	1
$k^{(0)}$	Penalty term in noise-aware loss	1
$\mathcal{K}_1$	Update $k^{(t)}$	0.1
$\mathcal{K}_2$	Update $k^{(t)}$	1
L	Maximum no. of the unlabeled samples	100,000

5 EXPERIMENTS

We seek to answer three key questions in our experiments. Q1. how does a LEAST-trained model perform in zero-shot and few-shot extraction scenarios? Q2. how do LEAST-trained models perform against state-of-the-art baseline methods? Q3. what are the individual contribution of some of the key components of our workflow on extraction performance? We answer the first question by training two state-of-the-art web-extraction models using LEAST. We evaluate our performance on two publicly available datasets in Section 5.4.1. We answer the second question by comparing our performance against a number of strong baselines in Section 5.4.5, and the third question by performing an ablation study in Section 5.4.6.

5.1 Experiment Design

5.1.1 Dataset. We evaluate all competing models on two publicly available benchmark datasets – The Structured Web Data Extraction (SWDE) [22] and Web Structured Reading Comprehension (WebSRC) [10] dataset. The SWDE dataset contains 8 verticals – *auto, university, camera, movie, job, book, restaurant*, and *nbaplayer*. Each

vertical consists of 10 websites and thousands of detail-pages. Some of these verticals (e.g. *nbaplayer, auto)* have more page-overlap across websites than other verticals (e.g. *movie, university*). Similarly, some verticals (e.g. *job*) have more detail-pages with variable text elements than boilerplate text compared to other verticals (e.g. *auto*). The WebSRC dataset, on the other hand, contains 10 verticals – *auto, book, camera, game, jobs, movie, phone, restaurant, sports* and *university*. Each vertical contains a varying number of websites[2]. Each website is comprised of 400 detail-pages. Our objective is to extract a vertical-specific attribute-set from each vertical. For the SWDE dataset, we use the officially released version to identify the attribute-set for each vertical. For the WebSRC dataset, we use the 'what is' question answer pairs to construct vertical-specific attribute-sets. We refer interested readers to the original papers for more details on each dataset.

5.1.2 Dataset partition. Following prior works [27, 54], we train a transferable model by pretraining it on a *pretraining corpus* first, and then fine-tune [37] it on a *fine-tuning corpus*. We construct our pretraining corpus by randomly sampling 90% of detail-pages from 2 seed websites in each vertical. Only 9 pages from each website are human-labeled while the rest is unlabeled. Our *validation set* contains the rest of the pages from the seed websites. The pretraining corpus also contains unlabeled pages from the rest of the websites in each vertical (except the target websites). Our fine-tuning corpus, on the other hand, contains detail-pages from the target websites. Only k pages randomly sampled from each target website is human-labeled in the fine-tuning corpus while the rest is unlabeled. We report end-to-end performance for both zero-shot ($k = 0$), and few-shot extraction scenarios ($k = 5, 100$). The pretraining and fine-tuning corpus contain detail-pages from the same vertical. We evaluate our model on 3 randomly sampled websites from each vertical. These 3 websites act as target websites[3] in our setup. Our *test corpus* contains detail-pages only from the target websites barring the human-labeled pages used to fine-tune the model. We follow this partition for both datasets. We repeat our experiments with 5 different permutations of seed and target websites and report the average score for each vertical.

5.1.3 Evaluation metrics. Following prior works [22, 29, 54], we evaluate the extraction performance of all competing models using the *page-level F1-score*. Briefly, the page-level F1-score represents the harmonic mean of the precision and recall of the model on a detail-page. We evaluate the correctness of a predicted attribute value by comparing it against the human-annotated groundtruth. We use the evaluation script released by Zhou et al. [54] to compute the average page-level F1-scores for both datasets. We compute the average F1-score over all verticals to compare our performance against each baseline method. Besides extraction performance, we also compute the label-efficiency of a LEAST-trained model. We measure the label-efficiency of a model by comparing the number of human-labeled samples needed to train the same backbone model using transfer learning [54] to obtain a comparable extraction performance.

Table 2: extraction performance of LEAST-trained models on 2 seed websites in the pretraining corpus

Dataset	Model	Zero-shot (%)	5-shot (%)	100-shot (%)
SWDE	SimpDOM	**48.63**	**95.34**	**95.98**
	MarkupLM	**57.58**	**97.24**	**97.72**
WebSRC	SimpDOM	**45.70**	**90.10**	**91.95**
	MarkupLM	**52.25**	**92.60**	**94.75**

5.2 Backbone Models

We use two state-of-the-art models as backbone in our experiment.

5.2.1 The SimpDOM Model. It is a LSTM based model [54] that employs some heuristics to simplify the HTML-structure of a detail-page by pruning off redundant DOM-nodes containing obvious non-matches and boilerplate text as a preprocessing step. It subsequently encodes each DOM-node in this simplified page using a set of carefully designed features, and classifies them as an attribute to be extracted (or None). Following the original work, we pretrain this model on a supervised extraction task on our pretraining corpus, and fine-tune it on a supervised extraction task on human-labeled pages in our fine-tuning corpus. We use LEAST-training in both pretraining and fine-tuning phases of our training workflow. We provide an overview of the architecture and hyperparameter settings of this model in Appendix A.

5.2.2 The MarkupLM Model. It is a Transformer-based model [27] that encodes each token appearing on a detail-page using a combination of text-based & XPath-based embeddings and classifies each DOM-node using a fully-connected layer. Following the original work, we pretrain this model on our pretraining corpus using a number of unsupervised objectives, and then fine-tune it using a supervised extraction task on the human-labeled pages in our fine-tuning corpus. We use LEAST-training only during the fine-tuning stage. We provide an overview of the architecture and hyperparameter settings of this model in Appendix A.

Implementation details. We use the open-source LXML library to preprocess each detail-page in our pretraining and fine-tuning corpus. More specifically, we apply the tree-simplification strategy proposed by Zhou et al. [54] to prune-off redundant nodes and edges from the HTML structure of a web-page for both datasets. This includes boilerplate nodes whose values are constant across the website (e.g. footers, navigational content). We also trim node-texts with more than 15 words. We use the official implementation[4,5] of each model in our experiments. We train both models on a NVIDIA V100 GPU following the original authors' recommendations for optimal performance.

5.3 Baselines

We compare our extraction performance against a number of strong baselines. They are as follows.

[2] ranging from 2 to 14

[3] see Section 2 for qualitative difference between seed and target websites

[4] https://github.com/google-research/google-research/tree/master/simpdom

[5] https://aka.ms/markuplm

5.3.1 Web-Extraction using Overlapping Data . WEIR [6] is an unsupervised algorithm that aligns detail-pages from partially overlapping websites to infer a list of wrappers for overlapping entities. We bootstrap these mappings using human-labeled pages in the pretraining and fine-tuning corpus to infer wrappers for each attribute.

5.3.2 Visual Rendering-based Model . Render-FULL [22] is a supervised model that utilizes a number of visual features to encode pairwise relationship between text-spans appearing on a detail-page and classifies them as one of the attributes to be extracted. We pretrain this model on a supervised IE task defined on the pretraining corpus, and then fine-tune it on the fine-tuning corpus in a similar way.

5.3.3 Simplified DOM-Tree . SimpDOM [54] is a state-of-the-art web-extraction model that utilizes a LSTM-based neural network to classify DOM-nodes on a detail-page. It utilizes a number of heuristics to prune redundant nodes from each detail-page and computes a feature vector for each DOM-node on this simplified page. To obtain a transferable model, we pretrain it on a supervised IE task defined on the pretraining corpus and then fine-tune it on the the fine-tuning corpus in a similar way.

5.3.4 Relational Graph Convolutional Network . Following Lin et al. [29], we implement a relational graph convolution network that encodes both local and pairwise relationships of each DOM-node using convolutional features. To obtain a transferable model, we pretrain the network on a supervised IE task defined on the pretraining corpus first, and then fine-tune it on the fine-tuning corpus in a similar way.

5.3.5 Structure-Aware Pretrained Model . MarkupLM [27] is a state-of-the-art Transformer-based model that encodes each token appearing on a detail-page using a combination of text-based and XPath-based embeddings. To obtain a transferable model, we pretrain it using a number of unsupervised objectives on the pretraining corpus fist, and then fine-tune it using a IE task defined on the fine-tuning corpus.

5.3.6 Sequentially Pretrained Model . H-PLM [10] serializes each detail-page by combining HTML tags and text-spans to compute a distributed representation of each token. It uses a pretrained ELECTRA model [11] as its backbone to compute these distributed representations. To obtain a transferable model, we pretrain it on a supervised IE task defined on the pretraining corpus first, and then fine-tune it on the fine-tuning corpus in a similar way.

5.3.7 Attribute Extraction via Question Answering . AVEQA [50] formulates the extraction task as a question answering problem. It uses a BERT-based [14] encoder to compute a distributed representation for each text-span and identify attributes as delimited text-spans that represent answers to vertical-specific questions. To obtain a transferable model, we pretrain this model on a supervised IE task defined on the pretraining corpus, and fine-tune it on the fine-tuning corpus in a similar way.

5.3.8 Uncertainty-Aware Text Classification. UST follows the uncertainty aware classification scheme proposed by Mukherjee et al. [34] for our IE task. Following the original work, we serialize each web-page as a sequence of tokens and assign each token a weight using the Monte-Carlo dropout-based learning strategy. We incorporate this weight as a label uncertainty measure and train a BERT-based model on this re-weighted corpus. We pretrain this model on a supervised IE task defined on the pretraining corpus and fine-tune it on the fine-tuning corpus in a similar way.

Table 3: extraction performance of LEAST-trained MarkupLM model on the SWDE dataset with 2 seed websites in the pretraining corpus

Vertical	Zero-shot (%)	5-shot (%)	100-shot (%)
nbaplayer	84.08	99.80	99.85
auto	53.22	98.0	99.04
movie	30.01	94.15	94.50
university	61.90	99.50	99.50
camera	49.35	92.11	93.22
job	48.0	97.86	98.0
restaurant	55.65	98.52	99.27
book	78.50	98.05	98.38
Average	**57.58**	**97.24**	**97.72**

5.3.9 Large Language Model. GPT-Neo-1.3B [2] is an open-source, pretrained Transformer model [48] that replicates the GPT-3 architecture [7]. It has ≈1.3B trainable weights. We serialize each web-page as a sequence of tokens, segment it into chunks to adhere to the maximum input sequence length of 512, and feed each chunk to this model. We use the following prompt template to perform the IE task: "`The following text is an excerpt from a website about <vertical>. I want you to output the value of the <attribute> from this text.`". For k-shot extraction scenarios, we also feed k human-labeled attribute-values from different webpages as exemplars to the model.

5.4 Results and Discussion

5.4.1 Extraction performance. We report the extraction performance of two LEAST-trained models on both experimental datasets in Table 2. We obtain better extraction performance on both datasets with the larger MarkupLM model. On the SWDE dataset, we obtain an average F1 score of 48.63% using the SimpDOM model and 57.58% using the MarkupLM model with no human-labeled pages from the target websites in the training corpus. On the WebSRC dataset, we obtain an average F1 score of 45.70% using the SimpDOM model and 52.25% using the MarkupLM model. For both datasets, extraction performance improves with the number of human-labeled samples from each target website in the fine-tuning corpus. This is because the quality of the pseudo-labeled samples from the target websites improve as the model gets exposed to more human-labeled samples from those websites. Improvement in extraction performance, therefore plateaus after a point as the transferability of the model saturates after encountering sufficient human-labeled samples from the target website.

5.4.2 Vertical-specific performance. To obtain better understanding of a model's performance that has been trained with LEAST, we report extraction performance of both models on each vertical

of the SWDE dataset. The vertical-specific breakdown reveals that we obtain better performance on verticals where the amount of page-overlap across websites is high, e.g. *nbaplayer*. We observe a F1-score of 84.08% in the zero-shot extraction scenario on this vertical. For verticals with high number of variable text elements than boilerplate text (e.g. *job*), on the other hand, extraction performance improves faster with the number of human-labeled pages from each target website. For example, in the *job* vertical, we observe an improvement of average F1 score by more than 49% by increasing the number of human-labeled pages from each target website from 0 to 5 in our fine-tuning corpus.

5.4.3 Diversity of training samples. We investigate the role played by the diversity of samples in our training corpus on extraction performance by varying the number of seed websites in the pretraining corpus. Our results (see Fig. 4) show that increasing the number of seed websites improves average F1 score for both models, although the improvement gets smaller for larger values of k in k-shot extraction scenarios[6].

5.4.4 Label-efficiency. We measure the label-efficiency of a LEAST-trained model by comparing the number of human-labeled pages needed to train the same backbone model using naive transfer learning [29, 54] to obtain comparable extraction performance. We pretrain this baseline model on our pretraining corpus and then fine-tune it on our fine-tuning corpus (see Section 5.1.2). If the extraction performance of the baseline model trained this way is not comparable with the LEAST-trained model, we increase the number of human-labeled pages from each website in both the pretraining and fine-tuning corpus by 25. We keep increasing the number of human-labeled pages in our training corpus until we have obtained a comparable F1 score using the baseline model. Recall that SimpDOM uses human-labeled samples during both pretraining and fine-tuning phases, whereas MarkupLM uses human-labeled samples only during the fine-tuning phase if our training workflow.

Table 4: Number of human-labeled pages needed for comparable zero-shot performance on the SWDE dataset with detail-pages from 2 seed websites in the pretraining corpus

Dataset	Backbone	# Human-labeled pages	Saved (%)
SWDE	SimpDOM	18	91.50
	MarkupLM	18	76.0
WebSRC	SimpDOM	18	83.45
	MarkupLM	18	72.90

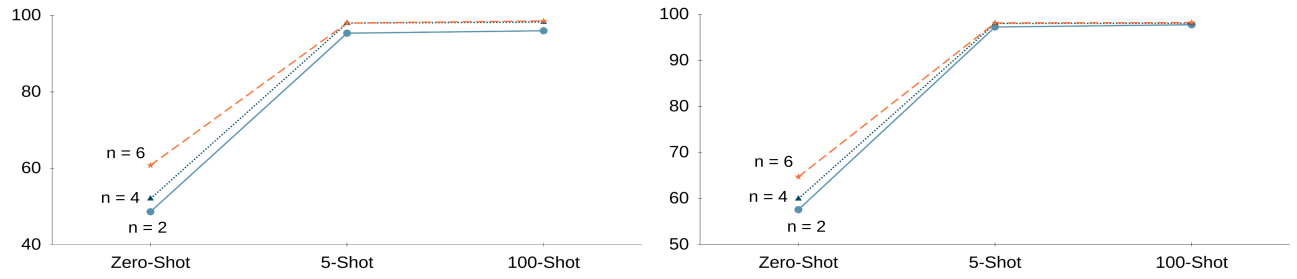
Let, the number of human-labeled pages required to train a model by following this approach for comparable extraction performance is $N_{\texttt{not\_LEAST}}$. The number of human-labeled pages needed to LEAST-train the same backbone model is $N_{\texttt{LEAST}}$. We define the label-efficiency of a LEAST-trained model as $\frac{N_{\texttt{not\_LEAST}} - N_{\texttt{LEAST}}}{N_{\texttt{not\_LEAST}}}$. We report the label-efficiency of both SimpDOM-LEAST and MarkupLM-LEAST for zero-shot extraction on the SWDE dataset and WebSRC dataset in Table 4. We observe that the SimpDOM-LEAST model requires 11.76x less human-labeled pages than its counterpart, whereas the MarkupLM-LEAST model requires 4.16x less human-labeled pages to achieve comparable performance. For the WebSRC dataset, the SimpDOM-LEAST model requires 6.04x less human-labeled pages than its counterpart, whereas the MarkupLM-LEAST model requires 3.69x less human-labeled pages. We observe similar trends for k-shot extraction scenarios as well. Percentage of human-labeled samples saved by the SimpDOM-LEAST model is comparatively larger than the MarkupLM-LEAST model for both datasets. This is because the SimpDOM-LEAST model is both pretrained and fine-tuned on human-labeled samples, and therefore has stronger correlation with the amount of human-labeled samples needed to train the model. The MarkupLM-LEAST model, on the other hand, only leverages human-labeled samples during the fine-tuning stage. LEAST reduces the number of human-labeled samples by a significant amount without trading off the extraction performance of both models.

[6] k = no. of human-labeled pages from a target website in the fine-tuning corpus

Table 5: extraction performance of a LEAST-trained SimpDOM model on the SWDE dataset with 2 seed websites in the pretraining corpus

Vertical	Zero-shot (%)	5-shot (%)	100-shot (%)
nbaplayer	82.41	99.71	99.83
auto	50.29	97.20	98.83
movie	24.65	92.12	92.02
university	30.50	99.98	99.98
camera	41.25	89.58	91.05
job	38.71	92.01	93.0
restaurant	48.40	97.22	97.90
book	72.85	94.90	95.27
Average	**48.63**	**95.34**	**95.98**

5.4.5 Comparison against baselines. We compare the extraction performance of LEAST-trained models on the SWDE dataset against a number of baseline methods in Table 6. For fair comparison, we follow the recommendations made by the respective authors for optimal performance. We use the same dataset partition for all competing models. Results show that LEAST-trained models outperform both WEIR and AVEQA in zero-shot and k-shot extraction scenarios. Extraction performance of WEIR suffers due to the limited number of human-labeled pages and marginal overlap across websites in many target verticals. The SimpDOM-LEAST model outperforms AVEQA by more than 9% on the zero-shot extraction scenario. Boost in performance is higher for k-shot extraction scenarios as the quality of the pseudo-labeled samples improves with the model getting exposed to more human-labeled samples from each target website. We also outperform the Render-FULL baseline by more than 28% on the zero-shot extraction scenario. From a qualitative standpoint, our method is more resource-efficient than Render-FULL as it does not require any of the computationally expensive processes (e.g. web-page rendering, visual feature computation) in its workflow. LEAST-trained models outperform the SimpDOM model. Leveraging additional training signals from the pseudo-labeled samples offers



(a) MarkupLM-LEAST **(b) SimpDOM-LEAST**

Figure 4: Average F1 score of LEAST-trained models on the SWDE dataset with samples from $n = \{2, 4, 6\}$ seed websites in the pretraining corpus

us a boost of more than 22% in extraction performance for the zero-shot scenario.

Table 6: Comparison of average F1 score of LEAST-trained models against baseline methods on the SWDE dataset

Model	No. of seed websites = 2		
	Zero-shot (%)	**5-shot (%)**	**100-shot (%)**
WEIR	8.12	9.05	14.27
Render-FULL	20.50	21.67	30.86
SimpDOM	25.98	27.02	38.95
RGCN	10.25	11.98	21.07
MarkupLM	41.67	43.59	92.01
H-PLM	38.30	41.88	88.72
AVEQA	39.18	41.02	86.50
UST	36.82	40.90	81.85
GPT-Neo	39.50	65.20	92.75
SimpDOM-LEAST	**48.63**	**95.34**	**95.98**
MarkupLM-LEAST	**57.58**	**97.24**	**97.72**

LEAST-trained models also outperform the RGCN baseline on both zero-shot and k-shot extraction scenarios. We obtain state-of-the-art results with the MarkupLM-LEAST model for the zero-shot extraction scenario, outperforming the H-PLM model by more than 19%, and the MarkupLM model by more than 15% in the zero-shot extraction scenario. We observe similar trend for the k-shot extraction scenarios as well. Comparing our models against the uncertainty-aware classification scheme undertaken in UST reveals improved extraction performance in all verticals. This is due to the quality of the pseudo-labels inferred by UST and its lack of structure-awareness in computing a distributed representation of each token. We outperfromed GPT-Neo on both zero-shot and 5-shot settings. On 100-shot setting, its performance improved as it was trained on more human-labeled examples. We hypothesize that there are two main reasons behind its poor performance on low-resource scenarios. *First,* it is not able to differentiate between structural tags and content tokens in the input sequence. *Second,* it is not able to identify and exclude boilerplate-text in the input sequence. Improved extraction performance of LEAST-trained models against mulitple strong baselines in both zero-shot and k-shot extraction scenarios establish the efficacy of our training workflow.

5.4.6 Ablation study. To understand the individual contribution of some of the key components in our workflow on extraction performance, we perform an ablation study in Table!7. We evaluate the contribution of a component by removing it from our workflow and compare the performance of the resulting model with a LEAST-trained model on the SWDE dataset. We perform these experiments using the MarkupLM model for the zero-shot extraction scenario. The second column in this table describes the component being removed, whereas the final column measures the drop in performance in the resulting model. In scenario A1, we remove the semi-supervised generative model (Section 4.1) from our training workflow. This affects the quality of the pseudo-labels in our training corpus. We observe a drop in performance of more than 15 F1 points, thus establishing the need of a supplementary supervision source when the number of human-labeled training samples in the training corpus is small. In scenario A2 and A3, we remove the re-weighting scheme and the noise-aware loss function from our workflow. We observe drops in extraction performance of 2.75 and 1.88 F1 points respectively. Finally, removing uncertainty-aware training from our workflow completely in scenario A4 results in a drop in extraction performance of 5.06 F1 points. This establishes the efficacy of uncertainty-aware training (Section 4.2) to ensure that our model does not degrade in quality due to error propagation from noisy pseudo-labeled samples.

Table 7: Results of ablation study

Index	Component ✗	ΔF1 score ↓ (%)
A1	Generative modeling for pseudo-labels	15.01
A2	Re-weighting	2.75
A3	Noise-aware loss	1.88
A4	Uncertainty-aware training	5.06

6 RELATED WORK

A. Web Information Extraction. Information extraction from web-pages is a well-studied problem [4, 46]. Traditional methods such as wrapper induction [20] require a large number of human-labeled pages from seed websites to learn custom rules or wrappers for each attribute. Fagin et al. [17] proposed a formal construct to represent extraction rules for identifying text spans in a document.

Although these methods yield high accuracy, they do not generalize well on websites that have varying page layouts. Due to the diverse nature of these pages, it is hard to scale these rule-based approaches for a large corpus as well. Zhai et al. [53] proposed an active learning-based approach to annotate additional pages from the target websites incrementally to modify the existing set of wrappers in a cost-effective way. Their approach however, require significant human-effort in terms of building specialized annotation tools, labeling new pages from each target website and more. Lockard et al. [30] proposed a distance supervision based approach to address this limitation. They used external knowledge-bases as a distant supervision source to annotate text-spans in a detail-page from target websites in a cost-effective way. Unfortunately, a comprehensive knowledge base may not always be available, especially for emerging verticals.

B. Label-Efficient Information Extraction. To improve the label-efficiency of web-extraction, Hao et al. [22] propose a set of weak but generalizable features to encode vertical-specific knowledge. Given an unseen target website, they apply these features to identify page-level candidates for each attribute first, and then remove spurious candidates by leveraging site-level information to boost extraction performance. The site-level information is derived in an unsupervised manner. Lockard et al. [31] encode contextual features for each DOM-node by using a graph neural network. To infer labels for a DOM-node appearing on an unseen target website, they leverage a semi-supervised label propagation algorithm. Wang et al. [50] formulates this extraction task as a question answering problem. Assuming each attribute to be extracted as the question, they develop a multi-task framework to identify text-spans corresponding to each attribute in a label-efficient way. The aforementioned works leverage a number of carefully designed visual features to encode a DOM-node. This has some serious implications on the efficiency of the IE workflow. Computing visual features from a fully rendered web-page is a continuous and resource-intensive process as it requires additional memory to store auxiliary files such as images, CSS, and JavaScripts that can easily get out-of-date.

C. Transferable Extraction Models. Lin et al. [29] develop a relational neural network that encodes pairwise relationship between neighboring DOM-nodes without requiring any visual rendering. They pretrain their model on a IE task defined on thousands of human-labeled pages from the seed websites and fine-tune it on the fine-tuning corpus. We compare our performance against a relational graph convolutional network in Table 6. We outperform this baseline on all verticals. Zhou et al. [54] follow a similar approach but improves the model's generalization capability on unseen target websites by introducing a tree-simplification strategy. They encode each DOM-node using a number of handcrafted features and classify them as one of the attributes to be extracted. We outperform this baseline on both zero-shot and k-shot extraction scenarios. Chen et al. [10] serializes a web-page as a sequence of tokens. They define the context of each token using both text as well as a sequence of HTML tags representing its XPath. They use a pretrained ELECTRA model [11] to classify each token as one of the attributes to be extracted. Wang et al. [50] improved the serialization aspect by introducing HTML tags as special tokens. They introduced cross-attention between text fields and HTML tokens within a web-page. Unfortunately, the aforementioned methods share a common flaw. Their transferability is correlated with the number of human-labeled pages used during pretraining. Quality of the model degrades when the number of human-labeled samples in the pretraining corpus is small. Recent works have addressed this by proposing unsupervised objectives to pretrain an extraction model. Both Deng et al. [13] and Chen et al. [9] pretrain their model on unlabeled detail-pages using masked language model (MLM) as their pretraining objective [14]. Li et al. [27] extend their work by proposing three pretraining objectives that adapt to the characteristics of semi-structured detail-pages. Leveraging these unsupervised objectives, these models can pretrain themselves on unlabeled web-pages in the pretraining corpus. Their end-to-end performance, however, still remains correlated with the number of human-labeled pages from the target websites used during the fine-tuning stage. We improve the label-efficiency of this model by more than $4x$ in our training workflow. We make similar observations for autoregressive large language models like GPT-3 [2]. Recent advances have brought these models to the forefront due to their capability of learning various downstream tasks from natural language text with limited human-labeled samples [7]. The corpus on which these models are pretrained are heavily biased towards natural language text, which affects their few-shot extraction performance from visually rich, complex documents. Recent works [1, 16] have established the necessity of high-quality, in-domain samples for robust extraction performance from such documents in few-shot scenarios. LEAST complements these models by constructing a pseudo-labeled corpus that can be used to fine-tune pretrained GPT-3 models with in-domain samples from the target vertical.

D. Self-Training for Classification Tasks. Semi-supervised models have been employed for many instance-level classification tasks in recent years [34, 47]. A naive extension of these techniques do not work for our task as they do not take the characteristics of semi-structured detail-pages into account. Self-training [45] has been a pioneer in terms of training a classification model on limited human-labeled samples [18]. Self-training with noisy pseudo-labels without degrading the model quality is still an active area of research. A majority of works [23] in this domain focuses on correcting the noisy labels by learning label corruption matrices. More related to our work, however, are instance re-weighting approaches [24, 34, 51] that down-weight samples that are more noisy. They rely on large language models and prior knowledge about the content of the pseudo-labeled samples to compute these weights. We do not rely solely on the content of a training sample to compute its weight. We also account for its source of origin. This makes our method more robust towards out-of-distribution samples in our training corpus. Experiments show that we outperform state-of-the-art uncertainty-aware classification models in both zero-shot and k-shot extraction scenarios.

7 CONCLUSION

Traditional web-extraction methods such as wrapper induction are hard to scale in scenarios where we want to extract from thousands of different websites in a vertical. If the layout of a target website is different from the seed websites used to learn these wrappers, additional human-labels may be required to modify them. Prior works have proposed transferable models to improve the label-efficiency

of training web-extraction models. Extraction performance of these models rely on the size of the fine-tuning corpus. The same is true for large language models like GPT-3 [2]. Although they generalize well for a wide range of natural language related tasks in few-shot settings [7], for visually rich [43] complex documents (e.g. semi-structured detail-pages), they need to be fine-tuned on in-domain, human-labeled samples [1, 16] for robust extraction performance. Constructing a large-scale fine-tuning corpus with samples from each target website, however, requires significant human-effort. We develop LEAST, a label-efficient self-training algorithm to address this gap. LEAST works in tandem with many publicly available models, including large language models, to construct a high-quality fine-tuning corpus with a limited number of human-labeled samples. To prevent the model from degrading in quality due to noisy training samples in the fine-tuning corpus, we develop an uncertainty-aware training strategy that estimates the label-uncertainty of each sample and then uses it to amplify (or dampen) the training signal of a sample during the gradient update step. Exhaustive experiments on two publicly available datasets show that LEAST generalizes to multiple target verticals. Using LEAST, we were able to train a transferable model with less than ten human-labeled pages from each seed website that outperformed state-of-the-art baselines by up to 22 F1 points in a zero-shot extraction scenario while reducing the number of human-labeled samples needed to train the same backbone model using a naive transfer learning algorithm by more than $11x$. Analyzing the performance of different models trained using LEAST with varying amounts of label-noise and understanding the impact of model architecture and pretraining strategies on their extraction performance requires a systematic study. It is one of our planned future works. Extending LEAST-training for large language models such as GPT-3 is another exciting direction of future work.

REFERENCES

[1] Monica Agrawal, Stefan Hegselmann, Hunter Lang, Yoon Kim, and David Sontag. 2022. Large language models are few-shot clinical information extractors. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. 1998–2022.

[2] Eleuther AI. 2021. *The GPT-Neo 1.3B model.* Accessed: 2023-04-05.

[3] Massih-Reza Amini, Vasilii Feofanov, Loic Pauletto, Emilie Devijver, and Yury Maximov. 2022. Self-training: A survey. *arXiv preprint arXiv:2202.12040* (2022).

[4] Mohd Amir Bin Mohd Azir and Kamsuriah Binti Ahmad. 2017. Wrapper approaches for web data extraction: A review. In *2017 6th International Conference on Electrical Engineering and Informatics (ICEEI)*. IEEE, 1–6.

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

[6] Mirko Bronzi, Valter Crescenzi, Paolo Merialdo, and Paolo Papotti. 2013. Extraction and integration of partially overlapping web sources. *VLDB* 6, 10 (2013), 805–816.

[7] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[8] Chia-Hui Chang, Mohammed Kayed, Moheb R Girgis, and Khaled F Shaalan. 2006. A survey of web information extraction systems. *TKDE* 18, 10 (2006), 1411–1428.

[9] Jingye Chen, Tengchao Lv, Lei Cui, Cha Zhang, and Furu Wei. 2022. XDoc: Unified Pre-training for Cross-Format Document Understanding. *arXiv preprint arXiv:2210.02849* (2022).

[10] Xingyu Chen, Zihan Zhao, Lu Chen, JiaBao Ji, Danyang Zhang, Ao Luo, Yuxuan Xiong, and Kai Yu. 2021. WebSRC: A Dataset for Web-Based Structural Reading Comprehension. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 4173–4185.

[11] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555* (2020).

[12] Wanyun Cui, Yanghua Xiao, Haixun Wang, Yangqiu Song, Seung-won Hwang, and Wei Wang. 2019. KBQA: learning question answering over QA corpora and knowledge bases. *arXiv:1903.02419* (2019).

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

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

[15] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *SIGKDD*. 601–610.

[16] Alexander Dunn, John Dagdelen, Nicholas Walker, Sanghoon Lee, Andrew S Rosen, Gerbrand Ceder, Kristin Persson, and Anubhav Jain. 2022. Structured information extraction from complex scientific text with fine-tuned large language models. *arXiv preprint arXiv:2212.05238* (2022).

[17] Ronald Fagin, Benny Kimelfeld, Frederick Reiss, and Stijn Vansummeren. 2015. Document spanners: A formal approach to information extraction. *Journal of the ACM (JACM)* 62, 2 (2015), 1–51.

[18] Benoît Frénay and Michel Verleysen. 2013. Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems* 25, 5 (2013), 845–869.

[19] Anna Lisa Gentile, Ziqi Zhang, and Fabio Ciravegna. 2015. Early steps towards web scale information extraction with lodie. *AI Magazine* 36, 1 (2015), 55–64.

[20] Pankaj Gulhane, Amit Madaan, Rupesh Mehta, Jeyashankher Ramamirtham, Rajeev Rastogi, Sandeep Satpal, Srinivasan H Sengamedu, Ashwin Tengli, and Charu Tiwari. 2011. Web-scale information extraction with vertex. In *ICDE*. IEEE, 1209–1220.

[21] Yu Guo, Zhengyi Ma, Jiaxin Mao, Hongjin Qian, Xinyu Zhang, Hao Jiang, Zhao Cao, and Zhicheng Dou. 2022. Webformer: Pre-training with web pages for information retrieval. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1502–1512.

[22] Qiang Hao, Rui Cai, Yanwei Pang, and Lei Zhang. 2011. From one tree to a forest: a unified solution for structured web data extraction. In *SIGIR*. 775–784.

[23] Dan Hendrycks, Mantas Mazeika, Duncan Wilson, and Kevin Gimpel. 2018. Using trusted data to train deep networks on labels corrupted by severe noise. *arXiv:1802.05300* (2018).

[24] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. 2018. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *ICML*. PMLR, 2304–2313.

[25] D. Kreines and B. Laskey. 1999. *Oracle Database Administration: The Essential Refe.* O'Reilly Media, Incorporated. https://books.google.com/books?id=WVC-R0gdl0kC

[26] Nicholas Kushmerick. 2000. Wrapper induction: Efficiency and expressiveness. *Artificial intelligence* 118, 1-2 (2000), 15–68.

[27] Junlong Li, Yiheng Xu, Lei Cui, and Furu Wei. 2021. Markuplm: Pre-training of text and markup language for visually-rich document understanding. *arXiv preprint arXiv:2110.08518* (2021).

[28] Xinzhe Li, Qianru Sun, Yaoyao Liu, Qin Zhou, Shibao Zheng, Tat-Seng Chua, and Bernt Schiele. 2019. Learning to self-train for semi-supervised few-shot classification. *NeurIPS* 32 (2019), 10276–10286.

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

[30] Colin Lockard, Xin Luna Dong, Arash Einolghozati, and Prashant Shiralkar. 2018. Ceres: Distantly supervised relation extraction from the semi-structured web. *arXiv:1804.04635* (2018).

[31] Colin Lockard, Prashant Shiralkar, Xin Luna Dong, and Hannaneh Hajishirzi. 2020. ZeroShotCeres: Zero-shot relation extraction from semi-structured webpages. *arXiv:2005.07105* (2020).

[32] Ilya Loshchilov and Frank Hutter. 2018. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*.

[33] Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *52nd ACL*. 55–60.

[34] Subhabrata Mukherjee and Ahmed Awadallah. 2020. Uncertainty-aware self-training for few-shot text classification. *NeurIPS* 33 (2020).

[35] S.B. Navathe, W. Wu, S. Shekhar, X. Du, X.S. Wang, and H. Xiong. 2016. *Database Systems for Advanced Applications: 21st International Conference, DASFAA 2016, Dallas, TX, USA, April 16-19, 2016, Proceedings, Part I.* Springer International Publishing. https://books.google.com/books?id=Ka7WCwAAQBAJ

[36] Robert Ormandi, Mohammad Saleh, Erin Winter, and Vinay Rao. 2021. Webred: Effective pretraining and finetuning for relation extraction on the web. *arXiv preprint arXiv:2102.09681* (2021).

[37] Sinno Jialin Pan and Qiang Yang. 2009. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* 22, 10 (2009), 1345–1359.

[38] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.
[39] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. 2018. Learning to reweight examples for robust deep learning. In *ICML*. PMLR, 4334–4343.
[40] Sebastian Ruder and Barbara Plank. 2018. Strong Baselines for Neural Semi-Supervised Learning under Domain Shift. In *56th ACL*. 1044–1054.
[41] Ritesh Sarkhel and Arnab Nandi. 2019. Deterministic routing between layout abstractions for multi-scale classification of visually rich documents. In *28th International Joint Conference on Artificial Intelligence (IJCAI), 2019*.
[42] Ritesh Sarkhel and Arnab Nandi. 2019. Visual segmentation for information extraction from heterogeneous visually rich documents. In *Proceedings of the 2019 international conference on management of data*. 247–262.
[43] Ritesh Sarkhel and Arnab Nandi. 2021. Improving information extraction from visually rich documents using visual span representations. *VLDB* 14, 5 (2021).
[44] Ritesh Sarkhel and Arnab Nandi. 2023. Cross-modal entity matching for visually rich documents. *arXiv preprint arXiv:2303.00720* (2023).
[45] Henry Scudder. 1965. Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory* 11, 3 (1965), 363–371.
[46] Hassan A Sleiman and Rafael Corchuelo. 2012. A survey on region extractors from web documents. *TKDE* 25, 9 (2012), 1960–1981.
[47] Antti Tarvainen and Harri Valpola. 2017. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *arXiv:1703.01780* (2017).
[48] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *NeurIPS* 30 (2017).
[49] Hongwei Wang, Fuzheng Zhang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2019. Multi-task feature learning for knowledge graph enhanced recommendation. In *WWW*. 2000–2010.
[50] Qifan Wang, Li Yang, Bhargav Kanagal, Sumit Sanghai, D Sivakumar, Bin Shu, Zac Yu, and Jon Elsas. 2020. Learning to extract attribute value from product via question answering: A multi-task approach. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 47–55.
[51] Yaqing Wang, Subhabrata Mukherjee, Haoda Chu, Yuancheng Tu, Ming Wu, Jing Gao, and Ahmed Hassan Awadallah. 2021. Meta Self-training for Few-shot Neural Sequence Labeling. In *SIGKDD*. 1737–1747.
[52] Sen Wu, Luke Hsiao, Xiao Cheng, Braden Hancock, Theodoros Rekatsinas, Philip Levis, and Christopher Ré. 2018. Fonduer: Knowledge base construction from richly formatted data. In *SIGMOD*. 1301–1316.
[53] Yanhong Zhai and Bing Liu. 2005. Web data extraction based on partial tree alignment. In *WWW*. 76–85.
[54] Yichao Zhou, Ying Sheng, Nguyen Vo, Nick Edmonds, and Sandeep Tata. 2021. Simplified DOM Trees for Transferable Attribute Extraction from the Web. *arXiv:2101.02415* (2021).

8 APPENDIX A

8.1 Overview of the SimpDOM Model

8.1.1 Architecture overview. SimpDOM is a LSTM-CNN-based model that represents each DOM-node on a detail-page using a combination of textual and discrete features. Fig. 5 shows an overview of its architecture. Given a detail-page, it first simplifies its HTML structure to extract contextual features for each DOM-node. For each node, it uses some heuristics to identify a subset of neighboring nodes as *friends* and *partners*. The textual representations for each node are then fed into a text encoder to generate a dense embedding. SimpDOM employs a hierarchical LSTM-CNN model to encode character-level and word-level features. It also incorporates discrete features computed from the markup information such as XPath, leaf node type, relative position, and pairwise semantic similarity to augment the node representation. This dense representation is then used to predict a softmax label for a node-text. We assign each DOM-node a label from the vertical-specific attribute set, or None. The model uses GloVe embeddings [38] to initialize the textual embedding of each node whereas other representations such as character embeddings and attribute embeddings are randomly initialized.

8.1.2 Training details. We determine optimal hyperparameters for both pretraining and fine-tuning phases by conducting a grid search. Following the original work, we set the dimension for both word embedding and character embedding to 100. We set the value of d_{path}, d_{pos}, and d_{leaf} to 30, 30, and 20 respectively. We use 50 filters and a 3×3 convolutional kernel for the CNN network. For the LSTM network, we set the hidden layer size to 100. We set both the maximum edge number and maximum ancestor number of a DOM-node to 5 and only keep the closest 10 *friends* of each node. We pretrain the model for 20 epochs and fine-tune it for 15 epochs with a batch size of 32. We use dropout with a probability of 0.3 to avoid overfitting. We use the Adam optimizer with a learning rate of 1e-3. We refer interested readers to the original work [54] for more background on this model.

8.2 Overview of the MarkupLM model

8.2.1 Architecture overview. MarkupLM is a Transformer-based [48] model that simultaneously encodes textual and XPath-based features of each token on a detail-page. Distinct from fixed-layout documents, web-documents use markup languages, e.g. HTML to encode two-dimensional position of each token in a web-page. MarkupLM takes advantage of the tree-based markup structures to model the relationship among different visual elements in the document. The model has four input embedding layers. The text embedding layer represents the token sequence information. The XPath embedding layer represents the markup tag sequence information (from root node to current node). The position embedding layer represents the relative sequence order information. The optional segment embedding layer is used by some downstream tasks. The overall architecture of the model is shown in Fig. 6.

8.2.2 Unsupervised pretraining. The MarkupLM model is pretrained on three unsupervised task objectives. They are as follows.

(1) The Masked Markup Language Modeling (MMLM) objective is designed to enhance the language modeling ability of the model. Given the text and markup input sequence of a detail-page, we randomly select and replace some tokens with the special token [MASK]. The objective of this task is to recover the masked tokens using contextual clues.
(2) Although the MMLM task helps the model improve its language modeling ability, the model is still not aware of the semantics of the XPath information encoded by the XPath embedding layer. The Node Relation Prediction (NRP) objective addresses this by encoding the relationship between a pair of DOM-nodes on a detail-page. We define a set of directed pairwise relationships $R \in$ {self, parent, child, sibling, ancestor, descendent, others}. Given a pair of nodes, the objective of this task is to predict the relationship (as defined in R) between them using only the first token of each node.
(3) For HTML-based documents such as detail-pages, the element <title> can be representative summaries of the element <body>. To utilize this self-supervised information, we randomly replace the text of the element <title> of a detail-page. Given the text of the element <body>, the objective of the Title-Page Matching (TPM) task is to predict if the title has been or not from the representation of the [CLS] token.

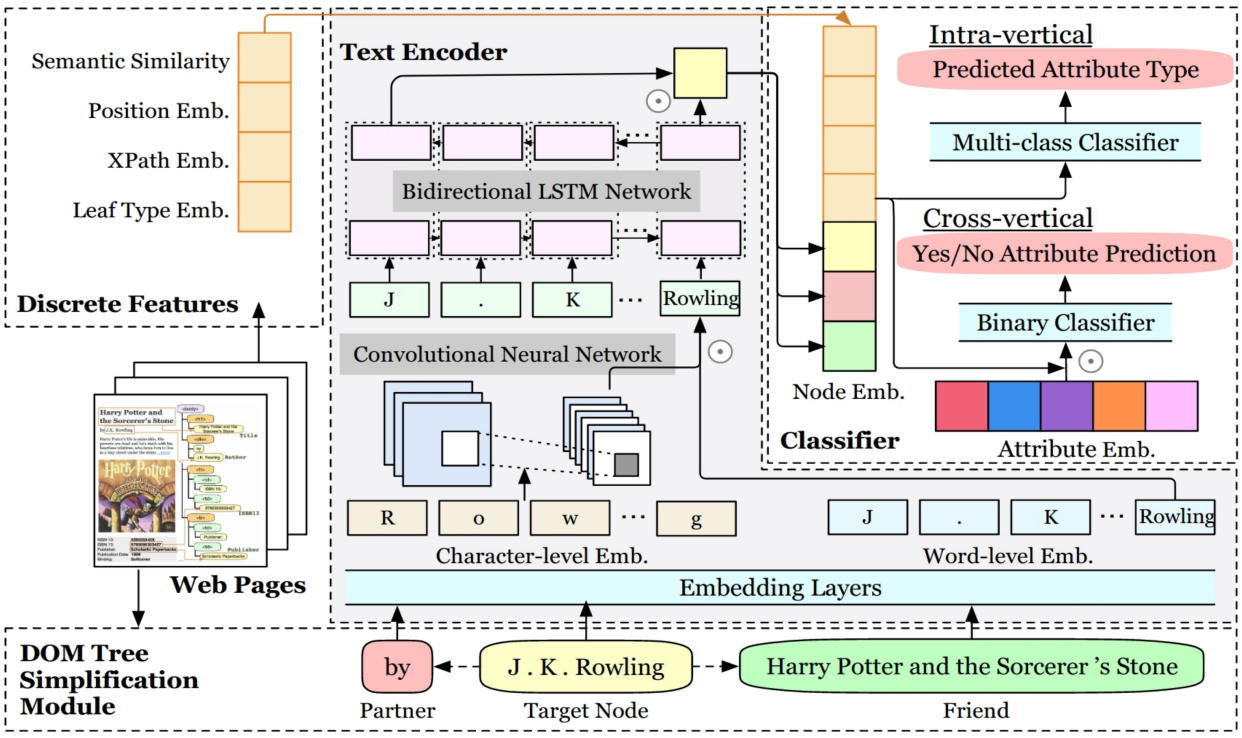


Figure 5: An overview of the SimpDOM model from Zhou et al. [54]. It encodes the textual features of each DOM-node on a simplified version of the detail-page shown in Fig. 1 using LSTM and CNN at the word-level and character-level respectively. A set of discrete features computed from the HTML structure of the web-page is also computed to compute a dense node representation.

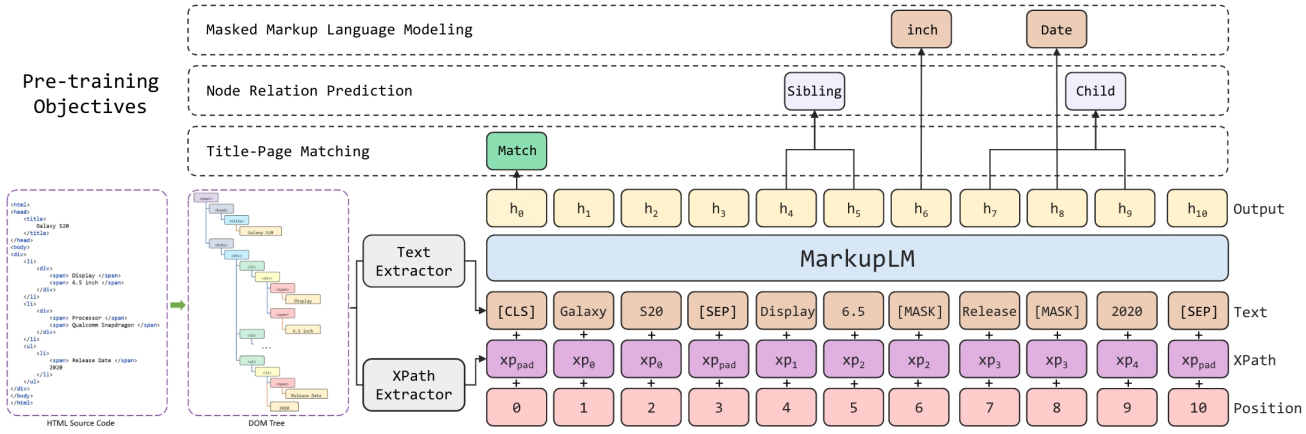


Figure 6: An overview of the MarkupLM model from Li et al. [27]. It uses three unsupervised task objectives to pretrain the model on the pretraining corpus and then fine-tunes it on a supervised IE task defined on the human-labeled pages of the target dataset.

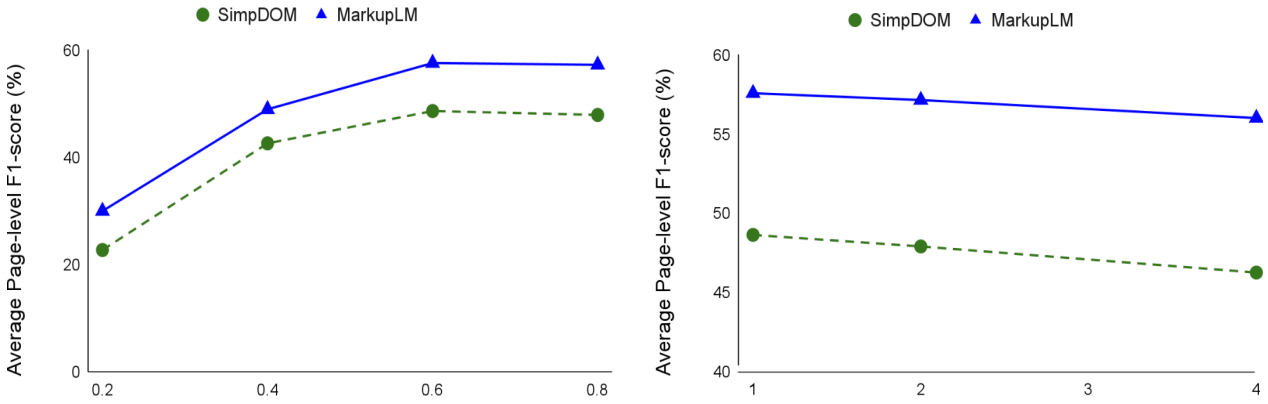
8.2.3 Training details. We set the size of the selected tags and subscripts in XPath embedding to 216 and 1001 respectively, the max depth of XPath expression to 50, and the dimension for the tag-unit and subscript unit embedding to 32. We set both the masking probability in the MMLM task and title replacement probability in the TPM task to 15%. We set the maximum number of node-pairs selected for the NRP task to 1000 for each page. We initialize the MarkupLM model with weights from a pretrained RoBERTa model and pretrain the model until convergence with a batch size of 32 and learning-rate of 5e-5. We use the AdamW optimizer [32] with a warmup ratio of 0.06, ϵ = 1e-6, β_1 = 0.9, β_2 = 0.98, weight decay = 0.01, and a linear decay learning-rate scheduler with 6% warmup steps for the pretraining phase. We fine-tune the MarkupLM model

for 15 epochs with a batch size of 32, learning rate of 2e-5, and warmup ratio of 0.1. We set the maximum sequence length to 384.

9 APPENDIX B

We present a sensitivity analysis of the hyperparameter $\beta^{(0)} \in (0, 1)$ for zero-shot extraction scenarios in Fig. 7.a. Recall that $\beta^{(t)}$ represents the probability of selecting the softmax label inferred by the semi-supervised generative model during pseudo-labeled corpus construction at iteration t. $\beta^{(0)}$ is the value that initializes this hyperparameter (Eq. 6). Fig. 7.a shows that for lower values of $\beta^{(0)}$, both model suffer in the zero-shot extraction scenario. This is because when initialized with a lower value, we end up preferring the softmax label inferred by the teacher model during pseudo-labeled corpus construction. As the number of human-labeled samples in our training corpus is small, a teacher model trained only on the human-labeled corpus can produce noisy pseudo-labels. This noise propagates through the model during training (Eq. 3) and degrades its extraction quality [40]. As the value of $\beta^{(0)}$ increases, preference shifts towards the semi-supervised generative model during the early iterations. We observe that the average page-level F1-score peaks near the value of $\beta^{(0)} = 0.6$ and then plateaus.

Fig. 7.b presents a sensitivity analysis of the hyperparameter $k^{(0)} \geq 1$. Recall that $k^{(t)}$ represents the penalty term that accounts for distribution shift during model training. $k^{(0)}$ is the value that initializes this hyperparameter (Eq. 5). Fig. 7.b shows that setting the value of $k^{(0)}$ to a higher value results in worse extraction performance. We observe that the average page-level F1-score peaks near the value of $k^{(0)} = 1$. This is because a higher penalty dampens the training signal from pseudo-labeled samples in the training corpus, which effects the quality of the model after an iteration and in turn the quality of the final model.



(a) Sensitivity analysis of the hyperparameter $\beta^{(0)}$

(b) Sensitivity analysis of the hyperparameter $k^{(0)}$

Figure 7: Sensitivity analysis of LEAST hyperparameters on the SWDE dataset for the zero-shot extraction scenario. Horizontal axis of each plot represents the value of the hyperparameter, whereas the vertical axis represents the average page-level F1-score.