Extracting Shopping Interest-Related Product Types from the Web

Yinghao Li[1*], Colin Lockard[2], Prashant Shiralkar[2], Chao Zhang[1]
[1]Georgia Institute of Technology, Atlanta, USA [2]Amazon, Seattle, USA
[1]{yinghaoli,chaozhang}@gatech.edu
[2]{clockard,shiralp}@amazon.com

Abstract

Recommending a diversity of product types (PTs) is important for a good shopping experience when customers are looking for products around their high-level shopping interests (SIs) such as hiking. However, the SI-PT connection is typically absent in e-commerce product catalogs and expensive to construct manually due to the volume of potential SIs, which prevents us from establishing a recommender with easily accessible knowledge systems. To establish such connections, we propose to extract PTs from the Web pages containing handcrafted PT recommendations for SIs. The extraction task is formulated as binary HTML node classification given the general observation that an HTML node in our target Web pages can present one and only one PT phrase. Accordingly, we introduce TRENC, which stands for **Tre**e-**Tr**ansformer **Enc**oders for **N**ode **C**lassification. It improves the inter-node dependency modeling with modified attention mechanisms that preserve the long-term sibling and ancestor-descendant relations. TRENC also injects SI into node features for better semantic representation. Trained on pages regarding limited SIs, TRENC is ready to be applied to other unobserved interests. Experiments on our manually constructed dataset, WEBPT, show that TRENC outperforms the best baseline model by 2.37 F_1 points in the zero-shot setup. The performance indicates the feasibility of constructing SI-PT relations and using them to power downstream applications such as search and recommendation.

1 Introduction

Customers of e-commerce websites fall in various stages of the purchase funnel[1] in their journey to purchase specific products. While lower-funnel customers target specific products or product categories, a customer in the middle to upper funnel only has vague shopping interests (SIs) and

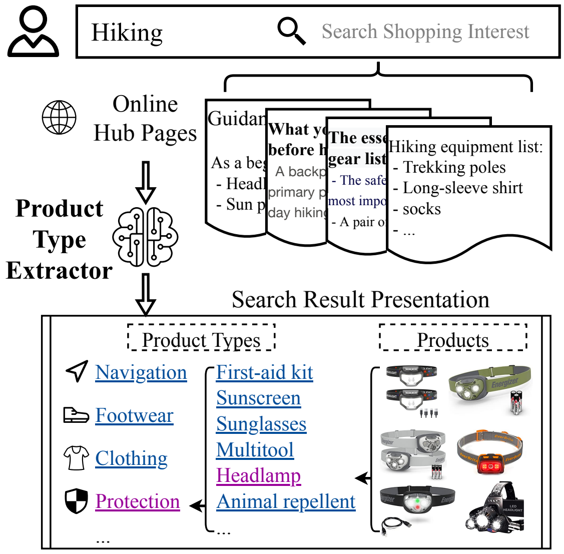


Figure 1: Example of the system and results to deliver as a response to searching for the SI "hiking".

requires additional guidance to determine the right products to purchase. Existing e-commerce websites are limited today in their ability to assist them in this kind of interest-oriented shopping. For example, a customer searching for *COVID-19 crisis* gets top results showing product types (PTs) such as books and test kits, while missing other essential categories such as the face mask, thermometer, or medicine. Moreover, the search result is a random assortment of products, without a clear organization that helps upper-funnel customers discover products within relevant categories.

The main problem is the concept of "shopping interest" is generally absent in e-commerce catalogs, which makes it difficult to directly establish the SI-PT connections and give corresponding recommendations. To circumvent such system limitations, customers today are accustomed to researching their products on hand-curated "hub Web pages"[2], each related to an SI and presenting PT

*All work performed while interning at Amazon.
[1]https://en.wikipedia.org/wiki/Purchase_funnel
[2]*E.g.*, hikinginthesmokys.com/hiking-checklist for hiking.

Findings of the Association for Computational Linguistics: ACL 2023, pages 7509–7525
July 9-14, 2023 ©2023 Association for Computational Linguistics

suggestions as organized lists, before returning to e-commerce websites. This stretches the total time spent on a purchase. We aim to find SI-related PTs directly on the e-commerce website, reducing customer effort for all their interest-oriented needs. Figure 1 shows the desired search experience.

The first step to this end is collecting hub pages, which is realized by querying Google Search with automatically selected prompts (appendix A). The rest of the paper focuses on PT extraction from the HTML pages, which presents several challenges. First, hub websites are heterogeneous in their format and terminology, with PTs often interspersed among long descriptive paragraphs, making it challenging for any solution designed for one or a few websites to work well for others. Second, our page collection approach assumes that all PTs presented on a page are related to the same SI, which may not hold true in practice, requiring us to filter out irrelevant PTs. Finally, our goal to find PTs for a wide range of SIs motivates us to consider a zero-shot learning setup (Xian et al., 2019) *w.r.t.* SIs, to generalize to interests not seen during training.

Representing an HTML document by a Document Object Model (DOM) tree whose nodes are HTML tags with text sequences, we formulate PT extraction as a node classification task that entails checking whether its text sequence represents a PT phrase. It is based on the empirical discovery that in our collected hub pages, a PT phrase generally occupies a single DOM node within a coherent group of enumerated HTML elements such as section titles or bullet points, where knowing one PT phrase suggests the potential presence of other PT phrases in the neighboring elements (Figure 3a). Node classification emphasizes learning inter-node structural dependencies rather than intra-node token interactions, which results in better generalization to a wide variety of HTML structures.

Due to the absence of a dedicated DOM tree encoding method, we propose TRENC (**Tr**ee-**Tr**ansformer **Enc**oders for **N**ode **C**lassification) to fill in the blanks. Adapted from the Transformer (Vaswani et al., 2017), TRENC incorporates ancestor-descendant and sibling node relations using modified self-attention mechanisms and positional embeddings that are suited to the unique DOM node arrangement of the rendered hub pages. The ancestor-descendant relation provides relative structural information between nodes in the DOM node hierarchy, whereas the sibling relation tracks the semantical connection among sibling nodes. The modified attention mechanisms reconstruct the tree architecture from the linearized input nodes and facilitate long-term dependency modeling. To capture the relevance between an SI and a node, we leverage a gating network to dynamically integrate SI semantics with a node's textual semantics, which generalizes TRENC to unseen SIs.

Evaluated on our dataset WEBPT with 453 Web pages covering 95 interests, TRENC achieves 2.37 absolute F_1 performance gain over the strongest baseline method. Our contributions include

- a novel and practical research topic of product type extraction from the Web pages associated with a given shopping interest;
- TRENC, a Transformer encoder-based model with structural attention mechanisms for recovering the DOM tree architecture from the node sequence to promote classification;
- a dataset WEBPT, and comprehensive evaluations of graph encoding techniques to verify the effectiveness of our model design.

The dataset is made publicly accessible at https://github.com/Yinghao-Li/WebIE to promote future research.

2 Related Works

Web Information Extraction Information extraction from the semi-structured Web data is a long-studied topic (Chang et al., 2006; Banko et al., 2007; Sleiman and Corchuelo, 2013). The works most relevant to ours are those on product attribute extraction (Zheng et al., 2018; Xu et al., 2019; Lockard et al., 2020; Zhou et al., 2021; Wang et al., 2022; Deng et al., 2022). For example, Zheng et al. (2018) train a BiLSTM-CRF network (Huang et al., 2015) for each attribute to locate its corresponding values on text sequences. Xu et al. (2019) scale it up by injecting the attribute name into the network as an attention objective. Wang et al. (2022) encode the DOM tree with graph attention network (Veličković et al., 2018) to incorporate the dependencies between nodes. However, attribute extraction is different from our PT extraction task at two major points. First, attributes are typically extracted from product detail pages, each of which mentions multiple attributes; and the attribute name-value pairs cluster around titles, bullet points and descriptions. In contrast, a hub page generally focuses on a single SI, with PTs scattered throughout the page. Unlike attribute extraction approaches that limit the searching scope to certain regions, the characteristics of hub pages require us to consider a page holistically instead of a small part. Second, attribute extraction is performed as token-level entity recognition in previous works, while PT extraction requires a node-level classification, which prevents approaches for the former from being directly applied to the latter. To our best knowledge, no applicable DOM node classification or similar dataset exists in openly available benchmarks such as OGB (Hu et al., 2020).

Graph Transformers Recently, graph neural networks (GNNs) such as the graph convolutional network (GCN, Kipf and Welling, 2017) and graph attention network (GAT, Veličković et al., 2018; Brody et al., 2022) have dominated the graph encoding research. But some works try to model graphs using Transformers (Dwivedi and Bresson, 2020; Maziarka et al., 2020; Ying et al., 2021; Park et al., 2022; Wu et al., 2022), to which our work is more related. For example, Maziarka et al. (2020) add inter-atomic distances into the self-attention heads to parameterize the molecular graph structure. Also targeting molecules, Graphormer (Ying et al., 2021) takes a step further and introduces centrality encoding, edge encoding and spacial encoding to evaluate the atom importance and capture the edge and graph structure. Park et al. (2022) and Wu et al. (2022) extend Transformers to knowledge graphs with partial message-passing tricks. Although applicable, the hierarchical and acyclic nature of DOM trees is different from the graphs for which the approaches were designed. Directly applying them to DOM trees leads to sub-optimal performance, as shown in § 5.

3 Problem Setup

We possess the DOM tree of a Web page associated with a given shopping interest C. The DOM tree can be represented by a set of nodes $\mathcal{V} = \{V_1, V_2, \dots, V_{|\mathcal{V}|}\}$ as well as a set of edges $\mathcal{E} = \{E_1, E_2, \dots, E_{|\mathcal{E}|}\}$ that connect the parent and children nodes. $|\mathcal{V}|$ and $|\mathcal{E}|$ are the sizes of node and edge sets respectively. We aim to design a binary node classifier $f : \mathcal{V} \cup \mathcal{E} \cup \{C\} \mapsto \{0, 1\}^{|\mathcal{V}|}$ to judge whether the text sequence in each node is a phrase representing a product type. The nodes with positive labels are referred to as "PT nodes" and the labels are denoted by $y_m = 1, m \in 1 : |\mathcal{V}|$. We focus our discussion on *one* DOM tree and use $m \in 1 : |\mathcal{V}|$ as its node index.

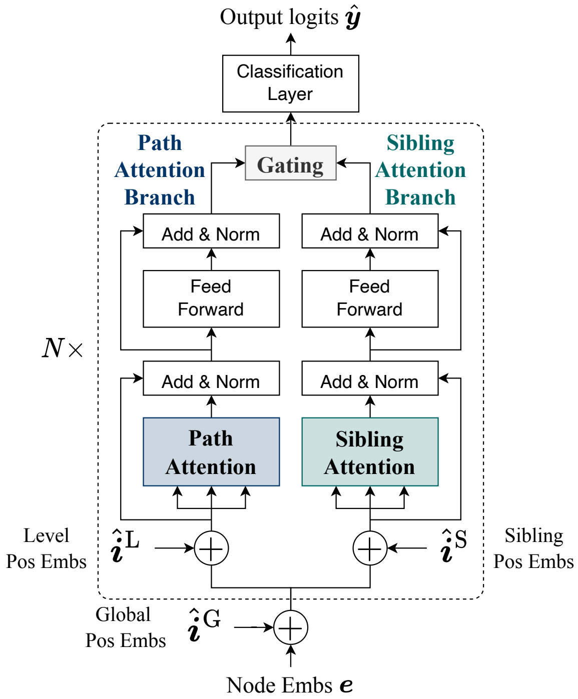


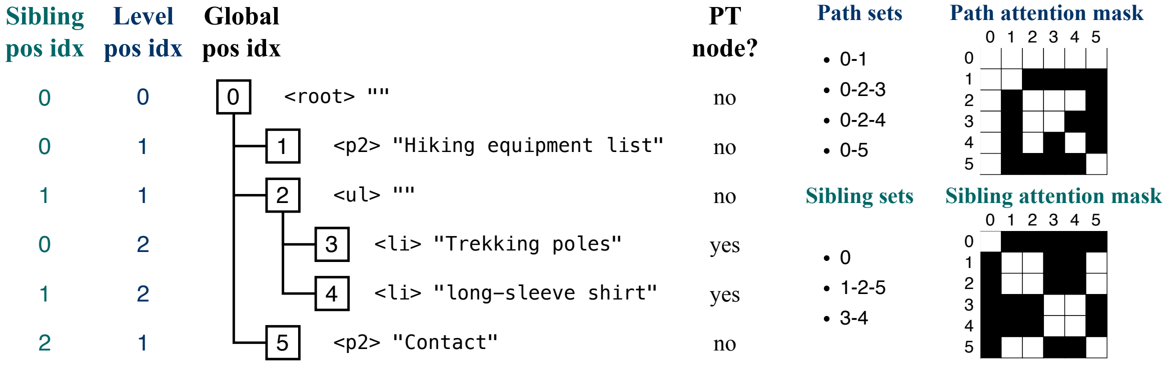
Figure 2: Model structure. It adopts path and sibling attention mechanisms to model the ancestor-descendant and sibling relationships among DOM nodes.

4 Method

We propose TRENC to model the DOM tree of hub Web pages for PT extraction. Figure 2 shows the model architecture. We treat the problem as a DOM node classification task that entails detecting whether its textual sequence defines a PT phrase. We first create a node representation that integrates three basic signals of a node that may be indicative of a PT (§ 4.1). We then adapt the Transformer architecture by adding two attention mechanisms, namely path attention and sibling attention, that allow capturing of inter-node dependencies presented by their HTML structure (§ 4.2.1). We also include three kinds of positional encodings that assist the attention layers with the node's unique positional information within the DOM tree (§ 4.2.2). Finally, we integrate the outputs from the path and sibling attention layers, which are used in a classification layer to predict node labels (§ 4.2.3). The implementation details are in appendix B.1.

4.1 Node Features

Besides the SI C associated with the tree, we consider two features for each node V_m: 1) its HTML tag $t_m \in \mathcal{T}$ where $\mathcal{T}$ is a finite tag set; and 2) the text sequence $\mathcal{S}_m = \{w_{m,1}, w_{m,2}, \dots, w_{m,|\mathcal{S}_m|}\}$, where $|\mathcal{S}_m|$ is the length and w is the token.



(a) Example of a simple DOM tree and corresponding positional indices

(b) Node sets and attention masks

Figure 3: An example of the model inputs, including the DOM tree components, positional indices and attention masks. The path sets and sibling sets in (b) are defined by the global positional indices in (a). In the attention masks, white elements have values 0 and the black ones are $-\infty$.

HTML Tag HTML tags are a finite vocabulary of keywords that define how browsers display their content. Specifically, they convey the semantic nature of their enclosed content. For example, <p> denotes a paragraph, while <ul> represents a list. Based on the observation that some tags tend to contain PT phrases more than others, we capture the tag information as a distinct structural feature and encode t_m with a vector $\boldsymbol{t}_m \in \mathbb{R}^{d_{\text{model}}}$ using an embedding layer. Here, d_{model} is the model dimensionality as in Transformers.

Text Sequence Text sequences convey the semantic character of an HTML document. In addition to directly indicating a PT phrase, they can also serve as useful contextual information about the neighboring nodes' propensity to contain a PT phrase. For example, a node stating "Essentials for camping" is a clear indicator that what follows is likely a set of camping-related PT nodes.

We leverage the power of pre-trained language models (PLMs) such as BERT (Devlin et al., 2019) to properly encode their semantics. For a given sequence, BERT generates an embedding $\boldsymbol{w}_{m,i} \in \mathbb{R}^{d_{\text{BERT}}}$ for each token $w_{m,i}, i \in 1 : |\mathcal{S}_m|$, besides two special tokens $w_{m,0}$ and $w_{m,|\mathcal{S}_m|+1}$ representing the start and end of the sequence. We derive the sequence embedding $\boldsymbol{s}_m \in \mathbb{R}^{d_{\text{model}}}$ by taking an average of all the token embeddings and passing it through a feed-forward network (FFN) layer:

$$\boldsymbol{s}_m = \boldsymbol{W}^{\text{seq}}(\text{GELU}(\frac{1}{|\mathcal{S}_m| + 2} \sum_{i=0}^{|\mathcal{S}_m|+1} \boldsymbol{w}_{m,i})), \tag{1}$$

where $\boldsymbol{W}^{\text{seq}} \in \mathbb{R}^{d_{\text{model}} \times d_{\text{BERT}}}$ are parameters.

Shopping Interest Although we assume that a DOM tree is associated with only one SI C, in rare cases this assumption does not hold. We are thereby motivated to capture the relevance between a node and the interest. Accordingly, we incorporate C with an embedding vector $\boldsymbol{c} \in \mathbb{R}^{d_{\text{model}}}$ in a similar manner as that for node text sequence (1), and let the model learn the relevance between C and related PTs to rule out any false positive cases.

Feature Integration We integrate node features into the node embedding $\boldsymbol{e}_m \in \mathbb{R}^{d_{\text{model}}}$ in two steps to honor the distinctiveness between the structural feature $\boldsymbol{t}_m$ and semantic features $\boldsymbol{s}_m$ and $\boldsymbol{c}$.

First, we merge the semantic features. Since different nodes have differing levels of correlations with the interest, we use gating vectors (Hochreiter and Schmidhuber, 1997) to automatically control how much interest embeddings $\boldsymbol{c}$ should be integrated into the sequence embedding $\boldsymbol{s}_m$. We calculate the weights $\boldsymbol{g}$ as:

$$\boldsymbol{g}(\boldsymbol{x}_1, \boldsymbol{x}_2) = \sigma(\boldsymbol{W}_1\boldsymbol{x}_1 + \boldsymbol{W}_2\boldsymbol{x}_2 + \boldsymbol{b}), \tag{2}$$

where $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are feature vectors; $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ are trainable square matrices; $\boldsymbol{b}$ is the bias, and σ is the sigmoid function. With (2), the updated sequence embedding vector becomes

$$\boldsymbol{s}'_m = \boldsymbol{g}(\boldsymbol{c}, \boldsymbol{s}_m) \odot \boldsymbol{c} + \boldsymbol{s}_m,$$

where $\odot$ is the element-wise product.

Then, we integrate the semantic and structural embeddings using concatenation followed by an FFN layer to maintain the embedding dimensionality. The integrated node embedding $\boldsymbol{e}_m$ is

$$\boldsymbol{e}_m = \boldsymbol{W}^{\text{emb}}[\boldsymbol{s}'^{\mathsf{T}}_m; \boldsymbol{t}^{\mathsf{T}}_m]^{\mathsf{T}},$$

where $[\cdot;\cdot]$ represents vector concatenation and $\boldsymbol{W}^{\text{emb}} \in \mathbb{R}^{d_{\text{model}} \times 2d_{\text{model}}}$ is an FFN layer.

4.2 TrENC Architecture

Compared with conventional GNNs that generally aggregate only 1-hop neighboring messages in each layer, Transformers are better at tracking long-term dependencies.However, applying the Transformer encoder to DOM trees as is can lead us astray because it is not designed to naturally accommodate the hierarchical structure of a tree. To address this limitation, we adapt the Transformer architecture by adding *structural attention* mechanisms with *node positional encodings* to better encode unique information within the DOM trees with the existing abilities of the Transformer architecture.

4.2.1 Structural Attentions

The DOM tree structure presents two kinds of relations that convey how nodes are related. The ancestor-descendant relation, represented by the edges $\mathcal{E}$, conveys the granular nature of a node (high or low) within the DOM hierarchy. The sibling relation between nodes conveys how they semantically represent a coherent group, as shown in Figure 3a. We incorporate these relationships via structural attention mechanisms, namely path attention and sibling attention. Correspondingly, we represent these two views of the DOM tree by two types of node sets: *path node sets* and *sibling node sets*. A path set $\mathcal{N}^{\text{P}} \subset \mathcal{V}$ is the ordered collection of all nodes in an HTML path, from the root node to an arbitrary node, as illustrated in Figure 3b. A sibling set $\mathcal{N}^{\text{S}} \subset \mathcal{V}$ consists of the immediate children of a non-leaf node. Thereupon, we develop *path* and *sibling attention* mechanisms, as described below, to explore the potential of modeling tree structures with Transformers.

Path Attention The path attention mechanism captures the granularity of a node V_m within the DOM tree, which carries useful information about the node's tendency to present a PT phrase. It limits the attention target of a DOM node to its ancestors or descendants only, echoing the edges $\mathcal{E}$ that define the DOM tree structure. Path node sets help define an attention mask toward this purpose by leaving out all "off-path" elements during the self-attention message passing operation.

Suppose the input is $\boldsymbol{H}^{\text{P}} \in \mathbb{R}^{|\mathcal{V}| \times d_{\text{model}}}$, in each attention head, the path attention scores $\boldsymbol{a}_m^{\text{P}} \in (0,1)^{1\times|\mathcal{V}|}$ of V_m attending to all DOM nodes are

$$\boldsymbol{a}_m^{\text{P}} = \text{SoftMax}(\frac{\boldsymbol{H}_m^{\text{P}} \boldsymbol{W}^{\text{Q}} (\boldsymbol{H}^{\text{P}} \boldsymbol{W}^{\text{K}})^{\mathsf{T}}}{\sqrt{d_k}} + \boldsymbol{M}_m^{\text{P}}). \tag{3}$$

Here $\boldsymbol{W} \in \mathbb{R}^{d_{\text{model}} \times d_k}$ are the FFN layers that map the latent features to the reduced d_k-dimensional single-head attention space, as in (Vaswani et al., 2017). $\boldsymbol{M}^{\text{P}} \in \{0, -\infty\}^{|\mathcal{V}|\times|\mathcal{V}|}$ is the path attention mask as shown in Figure 3b. $\forall u, v \in 1:|\mathcal{V}|$,

$$M_{u,v}^{\text{P}} = \begin{cases} 0, & \exists \mathcal{N}^{\text{P}} \textit{ s.t. } V_u \in \mathcal{N}^{\text{P}}, V_v \in \mathcal{N}^{\text{P}}; \\ -\infty, & \text{otherwise.} \end{cases} \tag{4}$$

$\boldsymbol{a}_m^{\text{P}}$ has non-zero values at positions corresponding to V_m's ancestors or descendants. The single-head attention output of V_m becomes

$$\text{Attn}_m^{\text{P}} = \boldsymbol{a}_m^{\text{P}} \boldsymbol{H}^{\text{P}} \boldsymbol{W}^{\text{V}}. \tag{5}$$

The rest of the architecture such as the layer norm and the residual connection is the same as in the Transformer and thus is omitted.

Sibling Attention Although sibling relations are not described by the edges $\mathcal{E}$, encoding them can provide a useful contextual signal based on the observation that sibling PT phrases often form a group. Accordingly, analogous to path attention, we develop sibling attention by imposing an attention mask $\boldsymbol{M}^{\text{S}}$, which forces a node to focus only on its siblings via self-attention. The sibling node set $\mathcal{N}^{\text{S}}$ helps define the mask. Its calculation is identical to (3)–(5), except that the variables are superscripted by sibling "$\cdot^{\text{S}}$" instead of path "$\cdot^{\text{P}}$".

4.2.2 Node Positional Encodings

Different from graphs, a DOM tree is acyclic and heterogeneous; the order of nodes influences their relations and how the elements are rendered. As Transformers do not encode such node order, positional embeddings are critical to capture such positioning. (Yun et al., 2020). We consider three types of absolute indices: *global*, *level* and *sibling* positional indices, as shown in Figure 3a. The global positional index i_m^{G} represents the position of each node in the tree in the depth-first order. It helps TrENC understand how the nodes are organized in the rendered HTML pages. The level index i_m^{L} and sibling index i_m^{S} on the other hand are developed to assist the path and sibling attentions. i_m^{L} describes the level or depth of a node, to help distinguish a parent from its children during the path attention, while the i_m^{S} captures the relative order among siblings within the sibling attention.

We encode positional indices by first applying sinusoid functions (Vaswani et al., 2017) to convert them to vectors $\boldsymbol{i}_m^{\mathrm{G}}, \boldsymbol{i}_m^{\mathrm{L}}, \boldsymbol{i}_m^{\mathrm{S}} \in [0,1]^{d_{\mathrm{model}}}$, followed by applying an affine transformation that maps each of them into distinct latent spaces:

$$\hat{\boldsymbol{i}}_m^{\mathrm{G}} = \boldsymbol{W}^{\mathrm{G}}\boldsymbol{i}_m^{\mathrm{G}}; \quad \hat{\boldsymbol{i}}_m^{\mathrm{L}} = \boldsymbol{W}^{\mathrm{L}}\boldsymbol{i}_m^{\mathrm{L}}; \quad \hat{\boldsymbol{i}}_m^{\mathrm{S}} = \boldsymbol{W}^{\mathrm{S}}\boldsymbol{i}_m^{\mathrm{S}},$$

where $\boldsymbol{W} \in \mathbb{R}^{d_{\mathrm{model}} \times d_{\mathrm{model}}}$ are FFN parameters.

4.2.3 TrENC Layers

In each layer, the path and sibling signals are modeled by two parallel branches, which are identical except for the positional embeddings and attention mechanisms (Figure 2). Denoting the input feature of layer l by $\boldsymbol{H}^{(l)} \in \mathbb{R}^{|\mathcal{V}| \times d_{\mathrm{model}}}$, we have[3]

$$\boldsymbol{H}_m^{\mathrm{P}} = \boldsymbol{H}_m^{(l)} + \hat{\boldsymbol{i}}_m^{\mathrm{L}}; \quad \boldsymbol{H}_m^{\mathrm{S}} = \boldsymbol{H}_m^{(l)} + \hat{\boldsymbol{i}}_m^{\mathrm{S}}, \quad (6)$$

which are passed into the attention sublayers (3)–(5) for message passing.[4] The branch outputs $\hat{\boldsymbol{H}}^{\mathrm{P}}$ and $\hat{\boldsymbol{H}}^{\mathrm{S}}$ are aggregated by a gating layer that generates the layer output $\hat{\boldsymbol{H}}^{(l)}$:

$$\begin{aligned}\hat{\boldsymbol{H}}_m^{(l)} =& \boldsymbol{g}(\hat{\boldsymbol{H}}_m^{\mathrm{P}}, \hat{\boldsymbol{H}}_m^{\mathrm{S}}) \odot \hat{\boldsymbol{H}}_m^{\mathrm{P}} + \\ & (\boldsymbol{1} - \boldsymbol{g}(\hat{\boldsymbol{H}}_m^{\mathrm{P}}, \hat{\boldsymbol{H}}_m^{\mathrm{S}})) \odot \hat{\boldsymbol{H}}_m^{\mathrm{S}}.\end{aligned} \quad (7)$$

The input of the first layer is the summation of the node embedding and global positional embedding $\boldsymbol{H}_m^{(1)} = \boldsymbol{e}_m + \hat{\boldsymbol{i}}_m^{\mathrm{G}}$, while the last output $\hat{\boldsymbol{H}}^{(N)}$ is fed into a classification layer to predict node labels, assuming the model has N layers in total.

4.3 Training and Inference

We use binary cross-entropy as our training objective. Suppose the predicted *logit* is $\hat{y}$, then the loss at the level of a DOM tree is calculated as

$$\ell = -\sum_{m=1}^{|\mathcal{V}|} y_m \log\sigma(\hat{y}_m) + (1-y_m)\log\sigma(1-\hat{y}_m).$$

During inference, we use 0.5 as a hard classification threshold for the predicted probability $\sigma(\hat{y})$.

5 Evaluation

In this section, we first describe a new dataset of interests and their associated webpages, specifically created to benchmark methods for the PT extraction problem. We then evaluate TrENC on the same, pitting it against a range of applicable baselines. Finally, we look at the effectiveness of various model components via ablation studies.

5.1 Experiments

Dataset We constructed a dataset containing 95 shopping interests and queried Google for hub pages using automatically selected prompts such as "[hiking] equipment list", where "hiking" is the SI. For each SI, we downloaded the top 100 returned pages and labeled them with PT nodes using a semi-automatic process. First, we applied simple heuristic rules to create noisy PT labels, based on structure and tag matching. Thereafter, for each SI, we presented roughly 5 webpages having a noisy label to a human annotator to further refine the labels. Even so, the dataset is not entirely noise-free given the subjective nature of the labeling process, with many ambiguous cases, such as deciding whether a software such as "VSCode" makes a valid product type. The pages without any positive human label were discarded. This process ultimately resulted in a collection of 453 HTML webpages having 94,167 nodes, among which 12,548 nodes are positive. Further details are described in appendix A.

Setup We focus on a zero-shot setup *w.r.t.* SIs since our goal is to evaluate various methods on SIs not seen during training. Therefore, we split the collection of webpages *stratified by their associated SIs* (recall that a webpage is assumed to be associated with only one SI) into training (75%), validation (10%) and test partitions (15%), ensuring that no SI is shared across partitions. As our dataset is small, we randomly split the collection 5 times and generated 5 distinct datasets, each with the three partitions. This approach is aimed to mitigate the impact of random factors while measuring real model performance. We identify the datasets as WebPT-n, where $n \in 1:5$ is the split index.

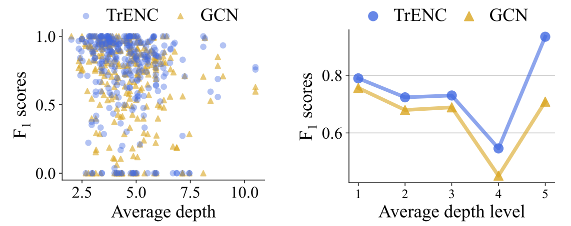
Baselines We consider the following simple to complex methods. 1) **Heuristic rules** are heuristic functions we manually designed to locate PT nodes from the DOM trees, which were also used to generate the initial, noisy node labels. 2) **Text similarity** decides whether a node is positive based on the cosine similarity between text and SI embeddings. 3) **Fine-tuned BERT** (BERT-FT) fine-tunes a BERT-base model to independently classify each tree node based on its text. 4) **Multilayer perceptron** (MLP) also classifies each node independently, but with

[3]Other positional encoding approaches such as (Chen et al., 2021) show similar performances.

[4]We omit the layer indicator $\cdot^{(l)}$ if possible for simplicity.

Models		WEBPT-1	WEBPT-2	WEBPT-3	WEBPT-4	WEBPT-5	$\bar{F}_1$ (precision / recall)
Heuristic Methods	Similarity Rules	40.12 56.53	39.14 62.44	35.84 56.90	36.55 59.68	33.80 58.28	37.09 (28.52 / 52.44) 58.77 (44.20 / **88.02**)
Supervised Methods	MLP	66.65	66.28	66.31	74.71	61.90	67.17 (72.11 / 63.38)
	BERT-FT	72.50	71.63	73.03	77.87	65.69	72.14 (68.32 / 76.65)
	Graphormer	71.09	81.76	75.73	66.81	69.67	73.01 (76.61 / 70.89)
	GAT	71.31	85.45	74.83	78.40	67.84	75.57 (77.07 / 74.28)
	GCN	76.13	84.07	**79.16**	81.50	71.92	78.56 (**84.44** / 73.57)
	TRENC	**79.65**	**88.26**	78.99	**82.40**	**75.35**	**80.93** (84.06 / 77.81)

Table 1: Test F_1 scores on each dataset WEBPT-n and the macro-averaged results (in %).



(a) F_1 *vs.* DOM tree depth (b) Average F_1 *vs.* depth level

Figure 4: Test F_1 scores against the DOM tree depths.

fixed BERT text embeddings followed by a set of FFN layers. 5) Graph neural networks (GNNs) propagate node semantics throughout the graph by aggregating neighboring node embeddings. GNN family has many variances, and we focus on **GCN** and **GAT**. 6) **Graphormer** (Ying et al., 2021), designed for molecular graphs, adds special encodings and attention masks to the Transformer model. Please see appendix B.2 for implementation details.

Metrics We evaluate each model with the F_1 scores corresponding to each split WEBPT-i and the macro-averaged F_1 score $\bar{F}_1 = \frac{1}{5}\sum_{n=1}^{5} F_{1n}$ with the corresponding macro precision and recall.

All trainable methods are equipped with early stopping techniques based on validation F_1 scores. To further reduce the influence of random factors without increasing training pressure, we store 5 snapshots of the models that perform the best on the validation dataset during training. During the test, we predict 5 sets of labels from the model snapshots and use the majority-voted labels as the final model predictions. It can be regarded as a simplified model ensemble method often used to improve model robustness (Dong et al., 2020).

5.2 Main Results

Table 1 shows the results of our comparative evaluation. As seen, TRENC outperforms all methods, exceeding the strongest baseline, GCN, by a margin of 2.37 absolute F_1 on average. Considering the small size of our datasets, it is not surprising that the test F_1 scores have relatively large variation across different data splits, as the correlation of data distributions of the training and test sets is susceptible to random factors. Nonetheless, TRENC achieves the best performance on 4 out of 5 splits as well as exceeds by a good margin on average, which strengthens the confidence of evaluation. Surprisingly, Graphormer underperforms GNN models and barely outperforms BERT-FT, a model that treats nodes independently without considering the tree structures. It indicates that models designed for other graphs such as molecular graphs are not directly applicable to our case. Instead of helping, the features Graphormer emphasizes *prevent* the model from learning a reasonable representation of the DOM tree. Table 1 also shows that the cosine similarity between SI and PT embeddings does not present a good performance. This is not unexpected as SI and PTs are not usually semantically similar, making it a sub-optimal way to directly compare their embeddings.

We also compare TRENC with GCN at varying levels of DOM tree complexity. Figure 4a shows tree-level F_1 scores of each DOM tree against its depth, which is the average depth of its nodes $\frac{1}{|\mathcal{V}|}\sum_{m=1}^{|\mathcal{V}|} i_m^{\mathrm{L}}$ and roughly echos the tree complexity. Figure 4b divides the depth equally into 5 levels and presents the average F_1 for each level. As seen, TRENC has better overall performance than GCN at all depths. In addition, the gap between TRENC and GCN increases when the tree is deeper, which indicates that TRENC can better encode complex trees due to the global message-passing ability of the self-attention mechanism.

5.3 Ablation Studies

We ablate input features and model components from TRENC to understand their effectiveness. Ta-

Models		Average F_1	Gap
TRENC		80.93	-
Input features	w/o SI	79.83	1.10 ↓
	w/o tag	78.82	2.11 ↓
	w/o text	57.98	22.95 ↓
Model components	w/o gating	80.39	0.54 ↓
	Transformer	78.44	2.49 ↓
	w/o pos emb	79.60	1.33 ↓
	w/o pth attn	78.98	1.95 ↓
	w/o sbl attn	78.04	2.89 ↓
Sequence encoding	BERT-large	79.36	1.57 ↓
	RoBERTa	78.54	2.39 ↓
	Sentence-BERT	74.73	6.20 ↓

Table 2: Ablation study F_1 scores (in %). "Pth" is short for path; "sbl" is short for sibling; and "pos emb" represents positional embeddings.

ble 2 shows the ablation results.

Input Features As seen, although removing any input feature (§ 4.1) impairs the model performance, text sequence is the most critical feature for TRENC. We further notice that without text sequence, TRENC performs quite close to the heuristic rules that utilize very limited lexical features (Table 1). This may indicate that TRENC exhausts the structural information available in a DOM tree.

Although not as significant as text sequences, incorporating SIs and tags does enhance the model performance. Injecting SIs turns the model's attention to their correlation with PTs. But such improvement is limited as the correlation is not strong, as discussed in § 5.2.

Model Components We investigate the functionalities of model components by removing them separately. The Transformer model discards edges $\mathcal{E}$ and treats the tree as a linearized sequence of nodes arranged by their global positional indices i^{G}. Although it learns certain structural dependencies, as indicated by its advance in comparison with MLP (Table 1), missing explicit edge knowledge still affects the model's judgment.

The results also show that path attention, sibling attention and positional encodings all contribute to better tree encoding. The row "w/o pos enc" removes the level and sibling encodings $\boldsymbol{i}^{\mathrm{L}}, \boldsymbol{i}^{\mathrm{S}}$ but keeps the global encoding $\boldsymbol{i}^{\mathrm{G}}$. Without $\boldsymbol{i}^{\mathrm{L}}$ and $\boldsymbol{i}^{\mathrm{S}}$, the model cannot properly identify the hierarchy and sibling order between nodes and therefore performs worse. Compared to path attention, sibling attention demonstrates a higher importance in context understanding, even though removing path attention means a node no longer has access to any other nodes from the tree.

	SI	Node text sequence
FP	at-home-spa	Esthetics or Skin Care
	hiking	Merrell Overlook Tall 2 WP Boot
	running	Credit card
FN	fishing	Rods for River Fishing
	canoeing	Water bottle - 1 litre is good

Table 3: Examples of common mistakes made by TRENC. FP/FN indicates false positives/negatives.

Sequence Encoding In our implementation, we use the uncased BERT-base model with $d^{\mathrm{BERT}} = 768$ as our encoders for sequence embeddings $\boldsymbol{e}$ and concept embeddings $\boldsymbol{c}$.[5] The embeddings are fixed during the training process.

We also test other pre-trained language models, including BERT-large, RoBERTa (Liu et al., 2019) and Sentence-BERT (Reimers and Gurevych, 2019), which is designed for comparing the sequence similarities and claims better sentence embedding performance than BERT. However, Table 2 shows that none outperforms BERT-base (Devlin et al., 2019). The reason might be the incompatibility of their training corpus and objective to our task. The results indicate that choosing an encoding model is vital to have good performance.

5.4 Case Studies on Classification Mistakes

Table 3 shows a few false positive (FP) and false negative (FN) examples to illustrate certain text sequence patterns where TRENC fails. As seen from FP cases, TRENC either struggles to determine whether it is a broad PT category (1st row), has challenges discerning a PT from a specific product (2nd row), or makes mistakes when unavoidable non-purchasable items are mentioned on the page along with other valid PTs (3rd row). From the FN cases, we conjecture that long descriptions may overwhelm the textual semantics and deviate its embedding, thereby preventing TRENC predict correctly (4th & 5th rows). The reason might be that TRENC have a stronger dependency on the node semantics than the structure, which is also indicated by the ablation results, and properly balancing the conditional terms may mitigate this issue.

[5] https://huggingface.co/bert-base-uncased

6 Conclusion

In this paper, we consider a new problem of extracting product types from the Web pages that are relevant to broad shopping interests such as camping. We model the problem as a node classification task and propose TRENC, a Transformer encoder-based model that leverages unique characteristics of DOM trees to perform product type extraction. In addition to the node-level signals including HTML tags, text sequences and shopping interest semantics, TRENC design path and sibling attention mechanisms based on DOM tree's ancestor-descendant and sibling relations. Together with the tree-based positional embeddings, the structural attention mechanisms promote the tree architecture understanding and make the classification more effective. Zero-shot experiments on a new dataset TRENC containing 95 shopping interests and 453 pages show that TRENC outperforms the baseline graph encoding models. This work pushes the frountier of researches of a more organized and intuitive result recommendation for middle-funnel customers.

Limitations

Apart from the issues mentioned in § 5.4, another limitation of TRENC is that it does not integrate any pre-training process such as BERT, which is effective in increasing the language understanding ability and adopted by previous works focusing on token-level classification tasks (Wang et al., 2022; Deng et al., 2022). Two factors lead to this decision. First, we use DOM nodes instead of tokens as the classification object and focus on relations between nodes rather than tokens. As the node text sequence is a composition of an arbitrary number of tokens, adopting the conventional masked language modeling (MLM) training objective (Devlin et al., 2019) seems impractical since there is *no direct mapping* from an embedding vector, one-hot encoded or not, to a sentence. The second reason is simply that we do not possess the corpus or computation resources for model pre-training. In fact, we expect a properly designed pre-training scheme to bring better node semantics representation and SI-PT relation modeling. It is an interesting topic and deserves further study.

Acknowledgments

This work was supported in part by Amazon.com Services LLC, NSF IIS-2008334, IIS-2106961, and CAREER IIS-2144338.

We would like to thank Xian Li, Binxuan Huang, Chenwei Zhang, Yan Liang, and Jingbo Shang for their insightful advice on this work.

References

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 2670–2676, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Shaked Brody, Uri Alon, and Eran Yahav. 2022. How attentive are graph attention networks? In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Chia-Hui Chang, M. Kayed, M.R. Girgis, and K.F. Shaalan. 2006. A survey of web information extraction systems. *IEEE Transactions on Knowledge and Data Engineering*, 18(10):1411–1428.

Pu-Chin Chen, Henry Tsai, Srinadh Bhojanapalli, Hyung Won Chung, Yin-Wen Chang, and Chun-Sung Ferng. 2021. A simple and effective positional encoding for transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2974–2988, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xiang Deng, Prashant Shiralkar, Colin Lockard, Binxuan Huang, and Huan Sun. 2022. DOM-LM: learning generalizable representations for HTML documents. *CoRR*, abs/2201.10608.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Xibin Dong, Zhiwen Yu, Wenming Cao, Yifan Shi, and Qianli Ma. 2020. A survey on ensemble learning. *Frontiers Comput. Sci.*, 14(2):241–258.

Vijay Prakash Dwivedi and Xavier Bresson. 2020. A generalization of transformer networks to graphs. *CoRR*, abs/2012.09699.

Matthias Fey and Jan E. Lenssen. 2019. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Colin Lockard, Prashant Shiralkar, Xin Luna Dong, and Hannaneh Hajishirzi. 2020. ZeroShotCeres: Zero-shot relation extraction from semi-structured webpages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8105–8117, Online. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Lukasz Maziarka, Tomasz Danel, Slawomir Mucha, Krzysztof Rataj, Jacek Tabor, and Stanislaw Jastrzebski. 2020. Molecule attention transformer. *CoRR*, abs/2002.08264.

Jinyoung Park, Seongjun Yun, Hyeon-Jin Park, Jaewoo Kang, Jisu Jeong, Kyung-Min Kim, Jung-Woo Ha, and Hyunwoo J. Kim. 2022. Deformable graph transformer. *CoRR*, abs/2206.14337.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 3980–3990. Association for Computational Linguistics.

Hassan A. Sleiman and Rafael Corchuelo. 2013. A survey on region extractors from web documents. *IEEE Transactions on Knowledge and Data Engineering*, 25(9):1960–1981.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations*.

Qifan Wang, Yi Fang, Anirudh Ravula, Fuli Feng, Xiaojun Quan, and Dongfang Liu. 2022. Webformer: The web-page transformer for structure information extraction. In *Proceedings of the ACM Web Conference 2022*, WWW '22, pages 3124–3133, New York, NY, USA. Association for Computing Machinery.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Qitian Wu, Wentao Zhao, Zenan Li, David Wipf, and Junchi Yan. 2022. Nodeformer: A scalable graph structure learning transformer for node classification. In *Advances in Neural Information Processing Systems*.

Yongqin Xian, Christoph H. Lampert, Bernt Schiele, and Zeynep Akata. 2019. Zero-shot learning — A comprehensive evaluation of the good, the bad and the ugly. *IEEE Trans. Pattern Anal. Mach. Intell.*, 41(9):2251–2265.

Huimin Xu, Wenting Wang, Xin Mao, Xinyu Jiang, and Man Lan. 2019. Scaling up open tagging from tens to thousands: Comprehension empowered attribute value extraction from product title. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5214–5223, Florence, Italy. Association for Computational Linguistics.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. 2021. Do transformers really perform badly for graph representation? In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 28877–28888.

Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank J. Reddi, and Sanjiv Kumar. 2020. Are transformers universal approximators of sequence-to-sequence functions? In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Guineng Zheng, Subhabrata Mukherjee, Xin Luna Dong, and Feifei Li. 2018. Opentag: Open attribute value extraction from product profiles. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, pages 1049–1058, New York, NY, USA. Association for Computing Machinery.

Yichao Zhou, Ying Sheng, Nguyen Vo, Nick Edmonds, and Sandeep Tata. 2021. Simplified dom trees for transferable attribute extraction from the web.

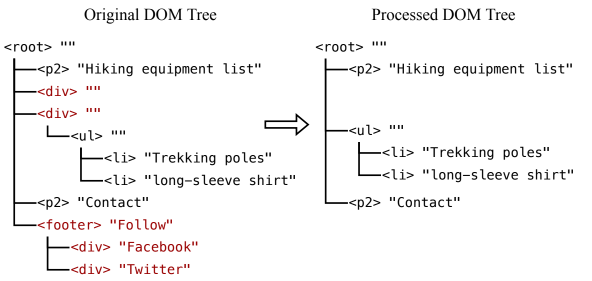


Figure 5: An example of HTML processing.

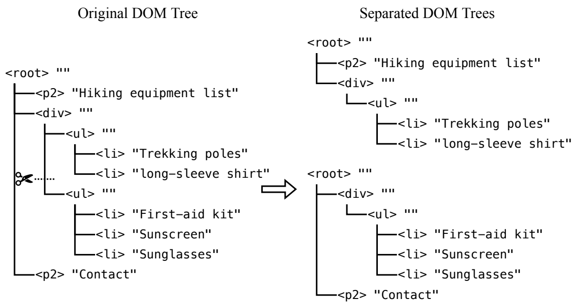


Figure 6: An example of separating a DOM tree.

A Dataset Details

A.1 Dataset Construction

We build WEBPT to realize a quantitative analysis of different PT extraction methods. WEBPT is a collection of hub pages relevant to a set of pre-defined SIs. Its construction process mainly consists of 5 steps: 1) defining SIs; 2) crawling hub pages; 3) processing HTML documents; 4) labeling documents; and 5) splitting data points.

Defining Shopping Interests As the first step, we establish a set of SIs through brainstorming. Particularly, we focus on popular activities, sports, hobbies and special events. Please check Table 6 and 7 for a complete list of SIs.

Crawling Hub Pages The hub pages are the webpages, each providing PTs related to a specific SI. Due to the variety of SIs, it is infeasible to focus on one or several websites for hub page collection. For example, a website specializing in sports will not provide information on "sewing" with a high chance and vice versa. In addition, gathering information from different websites may eliminate the bias probably existing in one website, according to the law of large numbers.

Considering this situation, we take advantage of Google Search with a simple query selector to locate the hub pages. Each SI C is combined with suffices "equipment list", "supply list", "tool list" and "checklist" before being fed into the search engine for querying. The system selects the combination with the largest number of results, whose top-100 query results are saved for later usage. We keep only the HTML pages and discard other documents such as PDFs or CSVs, so the actual number of saved documents may vary.

Processing HTML Documents This step aims to simplify the DOM tree structure to facilitate PT extraction. The RAW DOM tree is complicated with decorative and supporting scripts irrelevant to the content, which easily submerges the useful information we want to extract and decreases the false positive rate. We prune the trees by removing all headers, footers, and leaf nodes with empty text sequences. Then, we replace the nodes with only one child by their children to decrease the tree depth. To reduce the tree depth, we delete the nodes with only one child and then connect their children with subsequent subtrees directly with their parents. The process is illustrated in Figure 5. Experiments show that this HTML processing strategy successively simplifies the DOM structure without sacrificing any targeted content.

Labeling Documents and Splitting Data Points These two steps are sufficiently discussed in § 5.1 as will not be repeated. The only supplement is that the heuristic method used for initializing the noisy labels and compared in Table 1 is empirically developed. We omit its discussion since it is complex and not the focus of this paper. The detailed dataset splits are presented in Table 6 and 7.

Data Processing for Transformers One limitation of the Transformer models such as BERT and TRENC is that they need to set a constraint to the length of the input sequence $|\mathcal{V}|$ since the complexity of the self-attention mechanism is $\mathcal{O}(|\mathcal{V}|^2)$ and easily explodes when $|\mathcal{V}|$ is too large. Considering this drawback, for the node Transformers including Graphormer and TRENC, we set 512 as the maximum size of a DOM tree and split those that exceed this size. In addition, we guarantee that each split tree has 64 nodes at minimum. Figure 6 shows an example of the separation process.

A.2 Dataset Statistics

We present the dataset statistics in Table 4. DOM trees are larger than molecular graphs but significantly smaller than knowledge graphs.

Attribute	Value
# Shopping Interests	95
# DOM Trees	453
# Total Nodes	94,167
# Leaf Nodes	70,161
# Positive PT Nodes	12,548
Average # Nodes per Tree	207.87
Maximum # Nodes in a Tree	2,748
Minimum # Nodes in a Tree	19
Median # Nodes in a Tree	156
Average Tree Depth	7.06
Maximum Tree Depth	18
Minimum Tree Depth	3
Median Tree Depth	7
Average # Trees per SI	4.77
Average # Nodes per SI	991.23
Maximum # Nodes for an SI	3,050
Minimum # Nodes for an SI	363
Median # Nodes for an SI	935

Table 4: Dataset statistics.

A.3 Labeling Quality

The dataset is labeled by one individual as the task is straightforward. To investigate the labeling quality, we randomly select 25 DOM trees, removing their original labels and presenting them to 2 individuals for re-labeling. Table 5 presents the statistics and results. It shows that our labeling quality is decent despite some inevitable disagreements on ambiguous cases, as exampled in Figure 7.

Attribute	Value
# DOM Trees	25
# Total Nodes	5,938
# Positive PT Nodes	683
# Disagreement	87
# Fleiss' κ	98.53

Table 5: Annotation quality investigation.

Ambiguous Case 1

```
<root> ""
├──<p2> "Shirt"
└──<div> ""
    ├──<strong> "Men's shirt"
    ├──<span> "..."
    ├──<strong> "Women's shirt"
    └──<span> "..."
```

Ambiguous Case 2

```
<root> ""
├──<p2> "Shirt"
└──<div> ""
    ├──<p> "..."
    ├──<p> "..."
    ├──<strong> "shirt"
    └──<p> "..."
```

Figure 7: Examples of typical ambiguous annotation cases. In case 1, annotators may regard "men's shirt" and "woman's shirt" as negative as they are subcategories of the PT "shirt"; in case 2, the latter "shirt" may be regarded as negative as it is a repetition surrounded by long descriptive sentences.

A.4 Data Usage

All Web pages used by WebPT are included in the Common Crawl repository.[6] They are intended to provide information on a topic or interest, so consistent with that idea, we labeled the product types on each page. The labels do not contain any personally identifiable information. We are making the annotated dataset available to encourage further research on the product type extraction problem.

The WebPT dataset is licensed under the Creative Commons Attribution 4.0 International License. To view a copy of this license, visit http://creativecommons.org/licenses/by/4.0/.

B Implementation Details

B.1 TrENC Hyper-Parameters

We set the model dimensionality $d_{\text{model}} = 128$ and the number of TrENC layers $N = 12$. Each attention branch has 4 attention heads, and the single-head attention dimensionality $d_k = 32$. The feed-forward layer above the attention layer (Figure 2) first maps the features from d_{model} to a 512-dimensional latent space and then maps it back. The classification layer consists of 2 FFN sublayers that first downscale the TrENC layer output to 16-dimensional and then to the 1-dimensional output logits $\hat{y}$. We use the same activation functions and dropout strategy as described in (Vaswani et al., 2017). Our experiments show that the performance remains similar when we use 6 or 8 as the number of heads or use model dimensionality $d_{\text{model}} = 512$.

We train the model using 10^{-4} as the peak learning rate of the AdamW optimizer (Loshchilov and Hutter, 2019) with linear scheduler with 0.1 warm-up ratio. The batch size is 8 and the random seed is 42. We do not take multiple runs for each model on each dataset as our dataset and evaluation strategies (§ 5.1) can minimize the impact of random factors. Using another random seed (0) only changes the $\bar{\text{F}}_1$ scores of TrENC and GCN by 0.03 and 0.05, respectively. The model is implemented with the "Transformers" library (Wolf et al., 2020) in PyTorch (Paszke et al., 2019). The hyper-parameters not mentioned above keep their default values.

B.2 Baseline Methods

Text Similarity We adopt the same approach as described in § 4.1 with the uncased BERT-base

[6]https://commoncrawl.org/

model to generate the text sequence embedding $\boldsymbol{e}_m$ of each node V_m and the concept embeddings $\boldsymbol{c}$. Then, we compute their cosine similarity through

$$\mathrm{sim}_m \in (0,1) = \frac{\boldsymbol{e}_m^{\mathsf{T}}\boldsymbol{c}}{\|\boldsymbol{e}_m\|\|\boldsymbol{c}\|}.$$

We decide the classification threshold by exhausting possible values with 0.01 interval within $(0, 1)$ and select the one that gives the largest F_1 score. Notice that this threshold searching method is only applied to the text similarity baseline. Others take a constant threshold 0.5, as described in § 4.3.

BERT-FT BERT-FT classifies each node V_m independently by fine-tuning the uncased BERT-base model with the sequence classification task. The model input is the combination of the sequence $\mathcal{S}_m$ and the concept C, *i.e.*, "`[CLS]` $\mathcal{S}_m$ `[SEP]` C `[SEP]`". It does not consider the tag t_m. We append a one-layer FFN to the embedding corresponding to the `[CLS]` token to map it to a 1-dimensional logit. The training objective is minimizing the binary cross-entropy.

MLP MLP can be considered as a TRENC model without TRENC layers. In other words, it directly feeds the node embeddings $\boldsymbol{e}$ (§ 4.1) into the classification layer (Figure 2) without considering any inter-dependencies between nodes. We increase its classification layer depth until the validation F_1 stops improving for a fair comparison.

GNNs Similar to MLP, GNN models substitute the TRENC layers in the TRENC model with the GCN and GAT layers, respectively. The GNN layers are implemented with the "PyTorch Geometric" library (Fey and Lenssen, 2019). The number of GNN layers is fine-tuned according to the validation performance.

Graphormer We take the original implementation of Ying et al. (2021) and keep all model components.[7] The differences are that we initialize the node features with node embeddings $\boldsymbol{e}$ instead of atom categories, and we train the model with node classification instead of graph classification. We keep its scheme for encoding edges but introduce only one edge category representing the ancestor-descendant relationship.

SI	WEBPT-n				
	1	2	3	4	5
3d-printing	tr	tr	tr	tr	tr
airsoft-paintball	tr	tt	tr	tt	tr
archery	tr	tr	tt	vl	tr
astronomy	tr	tr	tr	tr	tr
at-home-fitness	tr	vl	tr	tt	tt
at-home-spa	tt	tr	tr	vl	tr
badminton	tr	vl	tr	tr	vl
baking	tr	tr	vl	tr	tr
bartending	vl	tr	tr	tr	tr
baseball	tr	tr	tr	tr	tr
basketball	tr	tr	vl	tr	tt
billiards-pool	tt	vl	vl	tt	tr
bird-watching	tr	tt	tr	tt	tr
boating	tr	tr	tt	tr	tr
bowling	tr	tt	tt	tr	vl
boxing	tr	tr	tr	tr	tr
calligraphy	tr	tr	vl	tr	vl
camping	tr	tr	tr	tr	tr
candle-making	tr	tr	tr	tt	vl
canoeing	tr	tt	tr	tr	tr
cheerleading	tr	tr	tr	tr	tr
cleaning	tr	tr	tr	tr	tr
climbing	tt	tr	tr	tr	vl
coffee	tr	tr	tr	tr	tr
comics-manga	tr	tr	tr	tr	tr
content-creation	tt	vl	tt	tr	vl
cricket	tr	tr	vl	tt	tr
crossfit	vl	tr	tr	tr	tr
cycling	tr	tr	tt	tr	tr
digital-art	tr	tr	tr	vl	tr
diy-home-improvement	tr	tr	tr	tr	tr
dj	tr	tr	tr	tr	tr
drag-queen	tr	tr	tr	tr	tt
drawing-and-sketching	tr	tr	tt	tr	tr
fencing	tr	tr	tt	tr	tr
field-hockey	tr	vl	vl	tr	tr
fishing	tt	vl	tr	tr	tr
floral-arranging	tr	tr	tr	tr	tr
football	vl	tr	tr	tr	tr
gaming	tt	tr	tr	tr	tr
gardening	tr	tt	vl	vl	vl
golfing	tr	tr	tr	tr	tr
gymnastics	vl	tr	tr	tr	tr
hair-care	tr	tr	vl	vl	tt
hiking	tt	tr	tr	tr	tr
hockey	tr	vl	tr	tr	tr
home-entertainment	tt	vl	tr	tr	tt
home-schooling	tr	tr	tr	tr	tr
horse-riding	tr	tr	tr	tr	tt

Table 6: Shopping interests and their splits in each dataset. "Tr", "vl" and "tt" represent "training", "validation" and "test" respectively.

[7] https://github.com/microsoft/Graphormer.

indoor-plants	tr	tr	tt	vl	tr
interior-design	tr	tr	tr	vl	tr
kayaking	tr	tr	tr	tr	vl
knitting	tr	tt	tr	tr	tr
lacrosse	tr	tt	tt	tr	tr
leathercraft	vl	tt	tr	tr	tr
makeup	tr	tr	tr	tr	tt
model-trains	tr	tt	tr	tr	tr
music-production	vl	tr	tr	tr	tr
nails	tr	tr	tt	tr	tr
painting	tr	tr	tr	tr	tt
paper-crafting	tr	tr	tr	tr	tr
parenting	tr	tr	tr	tr	tr
party-planning	tr	tr	tr	tr	tt
pet	tt	tr	tr	tt	tr
pilates	tr	tr	tr	tr	tr
pottery	tt	tr	tr	tr	tr
rugby	tr	tr	tr	tt	tr
running	tr	tt	tr	tr	tr
sailing	tr	tr	tr	tr	tr
scrapbooking	tr	tr	tr	tr	tr
scuba-diving	tr	tr	tt	tr	tt
sewing	tr	tr	tr	tr	tr
skating	tr	tr	tr	tr	tr
skiiing	tt	tr	tr	tr	tr
skin-care	vl	tr	vl	tr	tr
smart-home	tr	tr	tr	tr	tr
snowboarding	vl	tr	tr	tr	tr
soap-making	vl	tr	tr	tr	tt
soccer	tr	tr	tr	tt	vl
softball	tr	tr	tr	tr	tr
storage-and-organization	tr	tt	tr	tr	tt
student-dorm	tr	tt	tr	tr	tr
surfing	tr	tr	tr	vl	tr
swimming	tr	tr	tr	tt	tr
table-tennis	tr	tr	tt	tr	tt
teaching	tt	vl	tr	tt	tr
tennis	tt	tr	tr	tr	tr
travel	tr	tr	tt	tr	tr
volleyball	tr	tr	tr	tr	tr
weaving-and-spinning	tt	vl	vl	vl	tr
wedding	tr	tt	tr	tt	tr
wine	tr	tr	tr	vl	vl
work-from-home	vl	tt	tr	tt	tt
wrestling	tr	tr	tr	tr	tr
yoga	tr	tr	tt	tt	tr

Table 7: SIs and splits (cont.).

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Section 6

☒ A2. Did you discuss any potential risks of your work?
Our work is totally built upon publicly accessible materials. To the best of our knowledge, It will not trigger any ethical or safety concerns.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Section "Abstract" and section 1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

This topic is mainly discussed in appendix A and appendix B.

☑ B1. Did you cite the creators of artifacts you used?
Appendix B

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Appendix A.4

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Appendix A.4

☑ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Appendix A.4

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Appendix A

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Appendix A

C ☑ Did you run computational experiments?

Section 5

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Appendix B

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 5.1

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 5.2

□ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Not applicable. Left blank.

D ☑ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Section 5 and Appendix A

☒ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
The annotation guideline is straightforward and made orally.

☒ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No additional compensation was provided.

□ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
Not applicable. Left blank.

□ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Not applicable. Left blank.

□ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Not applicable. Left blank.