Webformer: Pre-training with Web Pages for Information Retrieval

Yu Guo*
Zhengyi Ma
Jiaxin Mao
Hongjin Qian
yu_guo@ruc.edu.cn
Gaoling School of Artificial Intelligence
Renmin University of China
Beijing, China

Xinyu Zhang
Hao Jiang
Zhao Cao
Distributed and Parallel Software Lab, Huawei

Zhicheng Dou†
Gaoling School of Artificial Intelligence
Renmin University of China
Beijing, China
Beijing Key Laboratory of Big Data Management and Analysis Methods
Beijing, China
dou@ruc.edu.cn

ABSTRACT

Pre-trained language models (PLMs) have achieved great success in the area of Information Retrieval. Studies show that applying these models to ad-hoc document ranking can achieve better retrieval effectiveness. However, on the Web, most information is organized in the form of HTML web pages. In addition to the pure text content, the structure of the content organized by HTML tags is also an important part of the information delivered on a web page. Currently, such structured information is totally ignored by pre-trained models which are trained solely based on text content. In this paper, we propose to leverage large-scale web pages and their DOM (Document Object Model) tree structures to pre-train models for information retrieval. We argue that using the hierarchical structure contained in web pages, we can get richer contextual information for training better language models. To exploit this kind of information, we devise four pre-training objectives based on the structure of web pages, then pre-train a Transformer model towards these tasks jointly with traditional masked language model objective. Experimental results on two authoritative ad-hoc retrieval datasets prove that our model can significantly improve ranking performance compared to existing pre-trained models.

CCS CONCEPTS

• **Information systems → Retrieval models and ranking**.

KEYWORDS

Ad-hoc Retrieval, Pre-training, Web Page, DOM Tree

*This work was done when Yu Guo was an intern at Huawei.
†Zhicheng Dou is the corresponding author.

SIGIR '22, July 11–15, 2022, Madrid, Spain

ACM ISBN 978-1-4503-8732-3/22/07…$15.00
https://doi.org/10.1145/3477495.3532086

ACM Reference Format:
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 (SIGIR '22), July 11–15, 2022, Madrid, Spain.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3477495.3532086

1 INTRODUCTION

In recent years, because of the powerful semantic representation and context modelling ability, pre-trained language models [7, 11, 28, 29] have been widely applied in the area of information retrieval [6, 18, 25, 38, 39, 46]. Recent studies have shown that applying these pre-trained language models (*e.g.*, BERT) can benefit the document ranking task and achieve better retrieval effectiveness [12, 24, 25, 39]. Furthermore, there have been some pioneer studies on designing the pre-training objectives tailored for IR [6, 17–20, 45]. For example, Ma et al. [18] proposed sampling word sets as pseudo queries based on a statistical language model. The pseudo queries are then used to construct query-document pairs which can fine-tune the PLMs dedicated for Information Retrieval. Lee et al. [17] and Chang et al. [6] proposed ICT, which sample a sentence in a paragraph as a pseudo query and regard the remaining paragraph as its corresponding document because of their strong correlation.

Although current IR-oriented PLMs have led promising improvements compared to using vanilla PLMs solely, there is still a large space for exploring better pre-training models for IR. In this paper, we reveal that there is a common problem in existing IR-oriented PLMs: they are trained with plain text and have ignored the useful structural information contained in large scale web pages. We argue that incorporating the structural information of web pages into PLMs would capture incremental semantic signals that can benefit the retrieval performance in IR. We think it is essential to investigate this problem because of the following reasons. **On the one hand**, there are a large amount of HTML web pages on the Web. They contain rich structural information organized by HTML tags together with plain text. For example, the texts in "title" tags

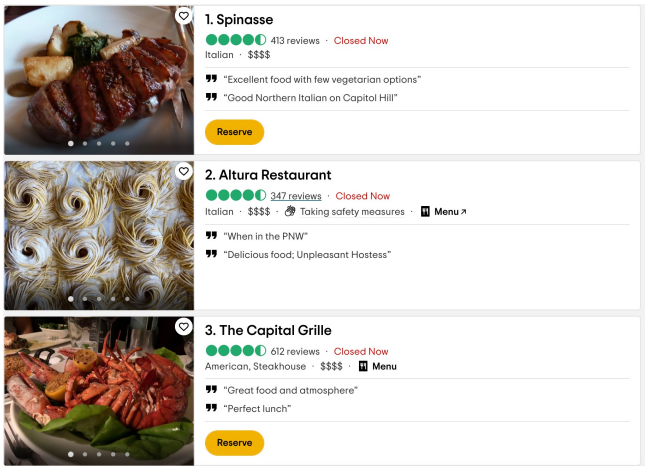

Figure 1: Illustration of structural information contained in a web page https://www.tripadvisor.com/Restaurants-g60878-zfp10954-Seattle_Washington.html. There are three similar content blocks in the web page. In each block, there are a title in bold and some descriptions to the restaurant, including location, comments and *etc.*.

are often the summarization of the document, and the parallel elements under the same parent tag usually share semantic relatedness with each other. Besides, web pages usually have regular shapes, as shown in Figure 1, we can see that the web page can be divided into 3 parts, and all the parts share the same structure with a restaurant name in bold and some descriptions to the restaurant in their fixed position, including location, comments and *etc.*. If we directly extract all the inner text, we will lose the information provided by the structure and have difficulty in understanding the texts' correlations. Intuitively, it is beneficial to consider the content structure of the HTML web pages for pre-training language models. **On the other hand**, the length limit of most PLMs are set as 512 tokens [11]. During pre-training, because the input is flattened text, they are unable to encode the whole web page at the same time thus simply truncate the texts or divide documents into passages. This will inevitably lose some important information for understanding the overall semantic information of the entire web page. Utilizing a new representation model that can naturally fit the hierarchical structure of web pages is hence essential.

Inspired by the two observations, we rethink the design of training objectives tailored for IR. In this paper, **we propose to pre-train language models with both the plain text and the structural information of web pages. Integrating the two kinds of complementary information, the PLMs can learn semantics from both the natural-language and the structure perspectives**. Leveraging structural information contained in HTML web pages has several advantages: (1) Since web pages are organized as HTML elements by HTML tags, language models can leverage the contextual information between HTML elements, thus build more accurate representations for texts and the whole document. (2) HTML elements usually contain multiple content blocks for specific information needs. Thus, they can reflect more precise and stable semantic information compared to fix-length text segments or chunks [9]. Leveraging HTML elements to split page content into passages is hence expected to enhance the quality of the pre-trained model. (3) HTML elements are organized as a DOM tree[1] in a hierarchical manner. Based on this structure, the model can learn long document text layer by layer instead of being limited to 512-token length. In this way, the language model can incorporate and preserve more information of one document while being able to model long documents naturally. (4) There are a huge number of pages on the Web, so data is sufficient if we want our model to scale up to billions of web documents.

In this paper, we propose a pre-training framework namely **Webformer**. We firstly construct a hierarchical model based on the structure of the DOM tree, and pre-train a language representation model with the supervised signals among structured HTML elements, then fine-tune the model parameters according to the downstream ad-hoc retrieval tasks. Considering the noise in original HTML text, we first remove uninformative text such as self-close tags, comments. Then, following Blohm [5]Mirończuk [21], we remove the internal attributes of the tags and treat every tag as a token. However, there are many non-leaf nodes in the web page's DOM tree that have only one child, which greatly increases the depth of the tree while little information is given. To address the huge overhead brought by the complex DOM tree structure, we compress the DOM tree to a definite depth. Specifically, if a non-leaf node has only one child node, its child node will be spliced to its parent node, and this non-leaf node will be deleted. After that, if the DOM tree's depth still exceeds maximum depth, the exceeded parts will be merged into its parent nodes. For example, if the maximum depth is 3, the "p" elements of the third layer in Figure 2 will become " textD " and become a leaf node. We encode the remaining nodes with two kinds of Transformer encoders namely Text Encoder and Node Encoder, where Text Encoder is used to encode leaf nodes and Node Encoder is used to encode non-leaf nodes. Because there are tag tokens in the leaf node, we pre-train a BERT-Base model from scratch with the HTML text using MLM objective to initialize Text Encoder. With such an approach, a whole web page can be naturally encoded by our model.

On the basis of the hierarchical model architecture, we design four self-supervised pre-training objectives for modeling the semantic correlations between HTML texts and hierarchical HTML elements in different views: (1) Masked Node Prediction (MNP). Inspired by the Masked Language Model (MLM), we treat all the child nodes of one node as a sequence, and predict the masked node's representation through the fine-grained information provided by its surrounding elements. (2) Parent-child Node Modeling (PNM). We use a non-leaf node as an anchor, and distinguish its child nodes from other irrelevant nodes, aiming to capture the correlation between parent node and its child nodes. (3) Sibling Node Modeling (SNM). Similar to PNM, we also distinguish sibling nodes from other irrelevant nodes, to model the parallel information in the web pages. (4) Children Order Prediction (COP). Similar to natural language sequence, the child nodes of one node also have a specific order. In order for the model to understand the relative order among nodes, we shuffle the child nodes' order and try to

[1]DOM Standard, https://dom.spec.whatwg.org/

predict the original order. Based on the four proposed tasks, we pre-train the Transformer model towards the supervised signals jointly with MLM objective. Via such a pre-trained method, Webformer can effectively fuse the structural and textual information in web pages, and learn context-aware language representations.

The pre-training dataset we use is English Wikipedia, which contains millions of well-formed Wikipedia web pages. At the fine-tuning stage, we use our pre-trained Text Encoder to initialize the ranking model, and evaluate its performance on two authoritative ad-hoc retrieval datasets, including the MS MARCO Document Ranking dataset [23] and the Trec 2019 Deep Learning dataset[8]. Experimental results show that Webformer trained on Wikipedia can outperform competitive state-of-the-art document ranking models. And we believe a larger corpus can release more potential power of Webformer in the future.

Our contributions are three-fold:

(1) We are the first to introduce structural information contained in web pages into pre-training for IR.

(2) We design a hierarchical Transformer model to encode long web pages, to capture the semantic correlations between HTML elements.

(3) We design four self-supervised pre-training objectives including Masked Nodes Prediction, Parent-Child Modeling, Sibling Nodes Modeling and Children Order Prediction, to exploit the structural information as supervised signals.

2 RELATED WORK

2.1 Pre-trained Language Models

Recent years have witnessed the great development of pre-trained language models. These context-aware representation models pre-trained on a massive amount of unlabeled data with deep neural networks have dominated a wide range of NLP tasks [11, 26, 41, 47]. Because of the powerful ability to aggregate context of fully-connected self-attention layers, Transformer [33] has become the main architecture of these pre-trained models. Devlin et al. [11] proposed BERT, a bi-directional Transformer model pre-trained with MLM and Next Sentence Prediction (NSP) to obtain contextual language representations and sentence-pair representations. Following BERT, many pre-training methods with different objectives have been designed, such as probabilistic language modeling [28, 40], permuted language modeling [41], sentence order prediction [16], and replaced token detection [7]. In order to encode long documents, some researchers explored how to increase the maximum length of the pre-trained model while preserving the its comprehension ability of textual information, and achieved excellent results [4, 15, 34, 42, 44]. The success of pre-trained models in NLP has also attracted much attention in the IR community. By feeding the concatenated query-document pair into BERT and simply adding an MLP layer to obtain a relevance score, many methods have achieved remarkable performance in fine-tuning for ad-hoc retrieval task [10, 12, 24, 25, 27, 32, 35, 39].

2.2 Pre-training Tasks for IR

In addition to general pre-training tasks, researchers found that adding good pre-training tasks that resemble the downstream tasks can obtain better performance at the fine-tuning stage [6]. And

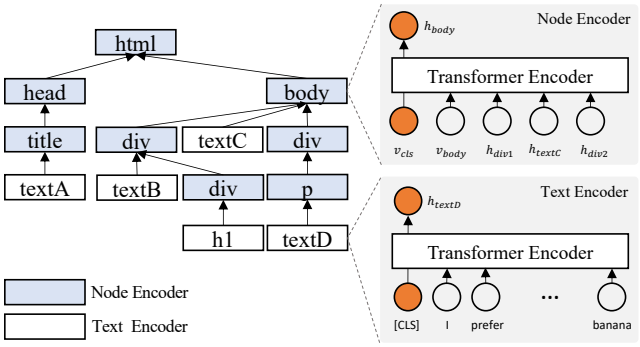

Figure 2: The structure of Webformer. A blue node represents the Node Encoder that encodes the information of a non-leaf node, whereas a white node represents the Text Encoder that encodes the HTML text of a leaf node.

the existing pre-training tasks such as MLM and NSP mostly focus on modeling the general contextual dependency or sentence coherence, not on measuring the query-document relevance, this inspired researchers to design pre-training objectives towards ad-hoc retrieval and collect corresponding pseudo query-document pairs. Lee et al. [17] and Chang et al. [6] firstly explored to pre-train the Transformer with Inverse Cloze Task(ICT) in dense retrieval task, where they treated the passages as the documents and the inner sentences as queries . Following ICT, Chang et al. [6] proposed Body First Selection (BFS) and Wiki Link Prediction (WLP) to model the inner page and inter-page semantic correlations. Ma et al. [18] proposed PROP with the Representation Words Prediction (ROP) task for ad-hoc retrieval. They sampled the word sets as pseudo queries based on a statistical language model, and assumed the query with a higher likelihood is more representative. After pre-training the Transformer model with ROP, they achieved state-of-the-art performance.

Different from the above approaches, we propose using the structure information contained in web pages comprised of HTML tags and elements as supervised signals for pre-training language models. HTML tags and elements have been leveraged in various existing works, including name entity recognition[3], zero-shot text summarization [1], document retrieval [2, 14], and information extraction [5, 21**?**]. However, none of the existing methods proposes to use HTML structural information to design pre-training objectives for IR. Since HTML is the source form of web pages and can provide stable complementary information to pure text, we believe they can bring more reliable supervised signals for pre-training, and further enhance the downstream IR task.

3 METHODOLOGY

In recent years, the PLMs have reformed the paradigm of the IR domain. However, most current PLMs are trained with plain text with general objectives (*e.g.*the MLM objective). In this paper, we seek to train the PLMs by leveraging the DOM trees of large-scale web pages and designing training objectives tailored for ad-hoc retrieval. We think the PLMs trained with our method would benefit

from the structural and hierarchical semantics of the web pages and therefore improve the performance of downstream tasks.

To achieve this, we design a model named Webformer. We train the Webformer via two stages: (1) pre-training stage and (2) fine-tuning stage. In the first stage, we use the DOM tree of the web pages to train the PLMs which is optimized with our designed objective functions. In the second stage, we fine-tune the PLMs obtained from the first stage with retrieval-task data (*e.g.*query-document pairs) to obtain the ranking model.

In this section, we will first provide an overview of our proposed model Webformer in Section 3.1, consisting of two stages of pre-training and fine-tuning. Then, we will give the details of the pre-training stage based on web pages in Section 3.2, and the fine-tuning stage for document ranking in Section 3.3.

3.1 Overview

We briefly introduce the two-stage framework of our proposed Webformer as follows.

3.1.1 Pre-training Stage. In the pre-training stage, we seek to train a transformer-based language model on the DOM tree of web pages. As shown in Figure 2, the DOM tree can be seen as a tree structure in which the HTML elements and text elements are the nodes of the tree.

Suppose that C is a large corpus of web pages, one web page $P \in C$ comprises of two kinds of nodes: non-leaf nodes E and leaf nodes L, *i.e.*, $P = (E, L)$. The non-leaf nodes $E = (E_1, E_2, \cdots, E_n)$ represent the collection of HTML elements, and the leaf nodes $L = (L_1, L_2, \cdots, L_m)$ represent the collection of text elements. For the i-th HTML element E_i, it might contain both HTML elements and text elements as the child nodes. For example, in Figure 2, the "body" element has three children including two "div" elements and one text node "TextC". For such a non-leaf node, we formulate the parent node and all of its child nodes into a heterogeneous sequence, *i.e.*, $E_{\text{body}} = (E_{\text{div1}}, L_{\text{TextC}}, E_{\text{div2}})$.

Besides, as mentioned in Section 1, in order to limit the model size, we set a maximum layer depth k of the DOM tree. If the depth of the DOM tree exceeds k, the parts exceeding layer k will be merged into the k-th layer. Based on the dataset C, we train a Node Encoder $\mathcal{M}^T$ and a Text Encoder $\mathcal{M}$ on this corpus.

Since the structural information in the web page is organized as DOM trees, we design a hierarchical framework to model the text and HTML elements of the web page. Specifically, as shown in Figure 2, We first expand each HTML file into a DOM tree, and then use two kinds of transformer encoders, *i.e.*, Text Encoder and Node Encoder, to encode the text and HTML element of the web page, respectively. For each node, we use the output of [CLS] of its corresponding encoder as its overall representation, and pass the vector to its parent node as a token of its parent's sequence. In this way, a piece of data starts from the leaf node, passes its semantic information step by step to the upper layer, and finally complete the modeling of the entire DOM tree.

To thoroughly capture the structural information of the DOM tree, we design four pre-training tasks from different perspectives of the HTML structural information of the web pages. Each pre-training task has a corresponding training objective. Our Webformer is trained based on the joint of the four pre-training tasks together with the MLM objective. Training with such a mechanism, we think the encoders can learn the structural information of web pages such as hierarchy and parallel, which facilitate the encoder to understand the semantics of web text better. Thus, it can achieve better performance when applied to the downstream task fine-tuning.

3.1.2 Fine-tuning Stage. To evaluate the effectiveness of our proposed Webformer, we fine-tune Webformer on the document ranking task. Specifically, for an ad-hoc query-document pair (q, d), we seek to learn a scoring model $\mathrm{s}(q, d)$ that measure the matching degree between the query q and the document d. We use the pre-trained Webformer as the backbone of the scoring model $\mathrm{s}(\cdot, \cdot)$. We concatenate the document d and the query q into a long sequence $[\mathrm{CLS}]d[\mathrm{SEP}]q[\mathrm{SEP}]$, and feed the sequence into the Webformer to obtain its hidden states. We use the representation of the $[\mathrm{CLS}]$ token to represent the whole sequence. The relevance score is computed by feeding the hidden states of the $[\mathrm{CLS}]$ token into a multi-layer perception (MLP).

3.2 Pre-training based on Web Pages

As we have introduced in Section 1, the information of web page is mostly organized as DOM tree in a hierarchical manner. However, the DOM tree of a web page contains both the HTML tag nodes and the text nodes, which have different semantic space. This heterogeneity inspires us to use two different Transformer encoders to encode the HTML tags and page texts, respectively. After generating the tag representations and text representations, we can leverage the supervised signals brought by HTML structural information to build reliable pre-training samples. To achieve this, we design four pre-training tasks based on HTML structural information to construct different loss functions. The architecture of the four pre-training tasks is shown in Figure 3. These four tasks try to learn the correlation between HTML tags and texts in different views. Thus, the focus of each task is how to build the pre-training sample. In the following, we will first present our hierarchical model for encoding the HTML tags and page texts, then introduce the proposed four pre-training tasks in detail.

The architecture for encoding the HTML elements and page texts is shown in Figure 2. We maintain a Text Encoder to encode the page texts in the web page, and use a Node Encoder to encode the HTML tags. Both the Encoders are Transformer encoders with the same architecture as BERT [11]. For the page texts $S = (w_1, w_2, \cdots, w_n)$ in the web page, we get its contextual representations by feeding the word sequence into the Text Encoder, and use the representation of the [CLS] token h_S as the text representation of S:

$$h_S = \mathrm{CLS}\left(\mathrm{TextEncoder}\left([\mathrm{CLS}], \mathrm{w}_1, \mathrm{w}_2, \cdots, \mathrm{w}_n\right)\right). \quad (1)$$

For the HTML tags containing plain texts, the representation of the inner texts is treated as this tag's representation, where inner texts is all the text included by this tag token. For the HTML tag containing multiple texts and tags, we use the Node Encoder to aggregate the children representations, and use the [CLS] representation as the tag representation. Specifically, for the tag containing multiple children $T = (t_1, t_2, \cdots, t_n)$, we calculate its representation h_T as:

$$h_T = \mathrm{CLS}\left(\mathrm{NodeEncoder}\left(v_{\mathrm{CLS}}, t_0, t_1, t_2, \cdots, t_n\right)\right), \quad (2)$$

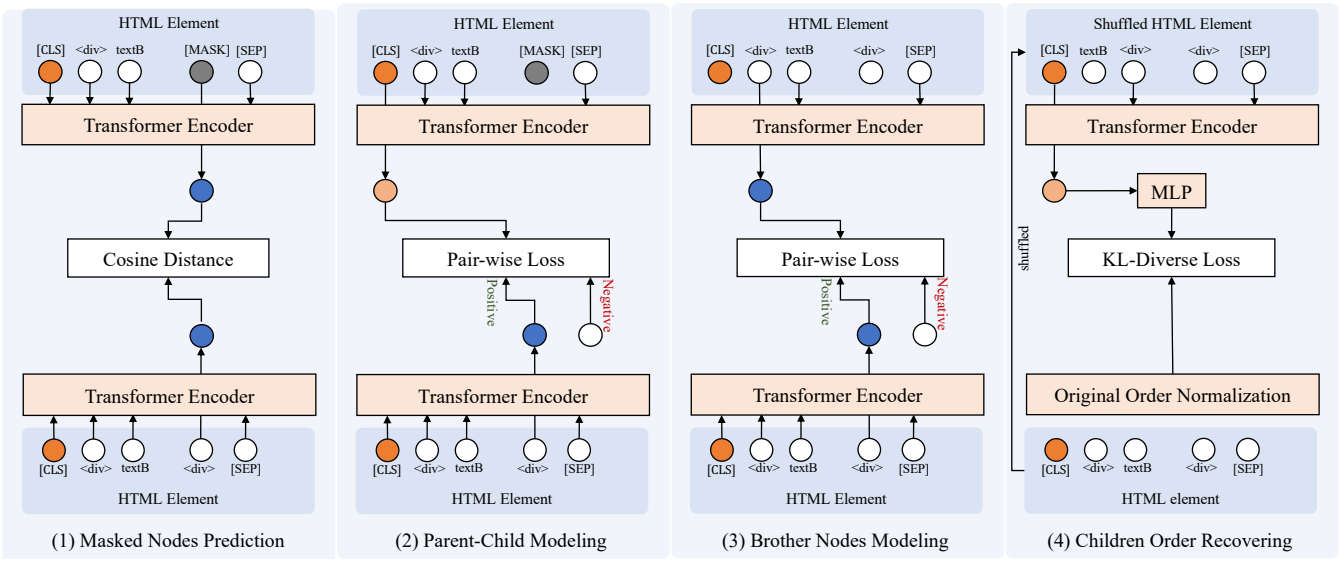

Figure 3: The proposed four pre-training tasks based on structural information in web pages: (1) Masked Nodes Prediction, (2) Parent-Child Modeling, (3) Sibling Nodes modeling and (4) Children Order Prediction

where v_{CLS} is the word embedding of [CLS] token and t_0 is the embedding of the tag token. In this way, we can calculate the representation of all nodes on the DOM tree, including the HTML tag representations and text representations. Since every representation is calculated based on its children, we can model and capture the structural information of the web page with the hierarchical model. After getting the encodings, we need to design specific pre-training tasks to optimize the encoder models for supervised training. In the following, we will present our proposed four pre-training tasks in detail.

3.2.1 Masked Node Prediction (MNP). In the DOM tree of a web page, the non-leaf HTML tags usually have multiple children nodes as their representation. For each child tag or child text, the surrounding elements can provide fine-grained information about it. Therefore, our first idea is to fuse the context elements through modeling the node correlations. In this way, it is expected to inject useful node correlations into tag representations and text representations. Inspired by the masked language model like BERT, we propose to model the context information in a node sequence by a cloze task, and try to approximate the masked node from the input sequence.

However, the MLM task has a specific label, which is the word that is masked. For a high-dimensional vector, we cannot provide an accurate supervision signal. In this regard, we reconstructed a new label, that is, we do not perform the masking operation on the input first, but after completing the bottom-up process of the model, we record the output of each step corresponds to the masked token as supervision signal. We hope to minimize the distance between the output of the masked position and the original output as much as possible.

Specifically, given a non-leaf node $T = (v_{\text{CLS}}, t_0, t_1, \cdots, t_n)$, we firstly use the Node Encoder to update its children's representations as $h_T = (h_1, \cdots, h_n)$. This representation is treated as the context-aware child nodes representations, since it is generated based on the bi-directional context of each nodes. Then, for the original $T = (v_{\text{CLS}}, t_0, t_1, \cdots, t_n)$, we randomly mask the j-th node t_j, and treat the rest sequence $T_{\text{mask}} = (v_{\text{CLS}}, t_0, t_1, v_{\text{mask}}, \cdots, t_n)$ as the surrounding context for t_j. We feed the surrounding context sequence T_{mask} into the Node Encoder, and calculate the hidden state of the t_j, then we extract the original context-aware representation of h_j in h_T, :

$$h_j^{\text{mask}} = \text{MASK}\left(\text{NodeEncoder}\left(T_{\text{mask}}\right)\right), \tag{3}$$

$$h_j = \text{MASK}\left(\text{NodeEncoder}\left(T\right)\right), \tag{4}$$

where h_j^{mask} is treated as the predicted representation based on its contexts, and $\text{MASK}(\cdot)$ is used to get the output of the [MASK] token. We formulate the objective of the Masked Node Prediction task by a cosine similarity-based loss. We minimize the cosine similarity between the predicted representation and the source representation of the masked node t_j as:

$$L_{\text{mnp}} = 1 - \frac{h_j^{\text{mask}} \cdot h_j}{||h_j^{\text{mask}}|| \cdot ||h_j||}. \tag{5}$$

3.2.2 Parent-Child Modeling (PCM). In the HTML DOM tree, most of the texts under the two nodes that construct parent-child relationship have obvious logical connections, since the information of the parent node is composed of all its children. Therefore, We hope to use such kind of relationship in the DOM tree to learn the containment relationship of the corresponding text. As mentioned above, in our model, we can use the aggregation result of child nodes to represent a non-leaf node, thus we can represent parent node and child node through dense vectors separately. We train the model to predict the true child node with the semantic information of the parent node, thus can learn the ability to distinguish containment relationships in pre-training. However, cause

we use the interaction results of all child nodes to represent the tag, there will be a problem of information leakage. Thus, after we select a child node, this child node is first removed from the DOM tree, and the representation of the parent node aggregated by the rest of the sequence is used to calculate the matching score with the original child node. In detail, for the input vector of a non-leaf node $T = (v_{\mathrm{CLS}}, t_0, t_1, t_2, \cdots, t_n)$, we randomly mask out the j-th vector and get $T_{mask} = (v_{\mathrm{CLS}}, t_0, t_1, v_{\mathrm{mask}}, \cdots, t_n)$ and pass it through the Node Encoder. The hidden state of the [CLS] represents the entire masked sequence, which is expressed as:

$$h_{\mathrm{parent}} = \mathrm{CLS}\left(\mathrm{NodeEncoder}\left(T_{\mathrm{mask}}\right)\right), \tag{6}$$

where $\mathrm{CLS}(\cdot)$ is a function to take the output of the [CLS] token. We use h_{parent} as the representation of the parent node, and the child node is the original masked vector t_j. After that, we randomly select a vector t_{sample} of another node representation in the same batch as a disturbance. We use $(h_{\mathrm{parent}}, t_j)$ as a positive example and $(h_{\mathrm{parent}}, t_{\mathrm{sample}})$ as a negative example to construct a pair-wise loss, which is expressed as:

$$L_{\mathrm{pcm}} = \max\left(0, 1 - \mathrm{P}\left(h_{\mathrm{parent}}|t_j\right) + \mathrm{P}\left(h_{\mathrm{parent}}|t_{\mathrm{sample}}\right)\right), \tag{7}$$

where $\mathrm{P}(x|y)$ is a score function,which can be expressed as:

$$\mathrm{P}\left(\mathrm{x}|\mathrm{y}\right) = \mathrm{MLP}\left([x; y]\right), \tag{8}$$

where [;] is a concatenation operation.

3.2.3 Sibling Nodes Modeling (SNM). Different from the containment relationship of parent-child nodes, sibling nodes contain more of a parallel relationship. It not only refers to the parallelism at the word level, but also includes parallelism between different areas of the web page. For example, the repeat regions on Figure 1 are all under the same "<table>" tag, and share the same content structure. In our model, two nodes with a parallel relationship are vectors that input to the same non-leaf node, thus we propose to use the representations to predict the true sibling node of one sampled node, thus can learn the ability to distinguish the parallel relationship between texts in pre-training. Specifically, for the input vector of a non-leaf node $T = (v_{\mathrm{CLS}}, t_0, t_1, t_2, \cdots, t_n)$, we first sample the j_1-th and j_2-th vector which is denoted as t_{j_1} and t_{j_2} separately, then, we randomly select a node representation t_{sample} of another piece of training data in the same batch. Finally, we use (t_{j_1}, t_{j_2}) as a positive example and $(t_{j_1}, t_{\mathrm{sample}})$ as a negative example to construct a pair-wise loss, which is expressed as:

$$L_{\mathrm{snm}} = \max\left(0, 1 - \mathrm{P}\left(t_{j_1}|t_{j_2}\right) + \mathrm{P}\left(t_{j_1}|t_{\mathrm{sample}}\right)\right), \tag{9}$$

where $\mathrm{P}(x|y)$ is the same as Equation (8).

3.2.4 Children Order Prediction (COP). In the text sequence, the relative order of tokens has strict requirements. Nevertheless, we want to argue that the relative order of regions of a web page also has strict requirements. Due to the length limitation of the previous pre-trained models, none of them can model multiple regions simultaneously, thus having difficulty in modeling the relative order among multiple regions. However, in our model, the child nodes of a non-leaf node represent areas that do not overlap each other, the deeper the node represents the larger the area. Inspired by this, we train the model to predict the original order of all the children of a non-leaf node, thus our model can recognize the relative order of different parts of a web page and have a clear understanding of their logical connections. Detailedly, for the input vector of a non-leaf node $T = (v_{\mathrm{CLS}}, t_0, t_1, t_2, \cdots, t_n)$, we keep the positions of v_{CLS} and t_0 unchanged, randomly shuffling the order of its rest child nodes and get $T_{\mathrm{shuffle}} = (v_{CLS}, t_0, t_{m_1}, t_{m_2}, \cdots, t_{m_n})$, where $(m_1, m_2, \cdots, m_n)$ is the original order of the child vector, the final after that, we pass the [CLS]'s output to an MLP to get a d-dimensional vector,then use softmax function to normalize the vector,which can be expressed as:

$$O = \mathrm{softmax}\left(\mathrm{MLP}\left(\mathrm{CLS}(T_{\mathrm{shuffle}})\right)\right) = (o_1, o_2, \cdots, o_d), \tag{10}$$

where d is the max length of children nodes. At the same time, the original order is expanded to d-dimension as $M = (m_1, m_2, \cdots, m_n, \cdots, d)$, where d is the maximum length of Node Encoder. We also normalize the vector as:

$$W = \frac{M}{\sum_{i=1}^{d} m_i}. \tag{11}$$

Finally, we use the KL divergence of these two distributions as the loss function of COP, expressed as:

$$L_{\mathrm{cop}} = \mathrm{D}_{\mathrm{KL}}(O||W) = \sum_{i=1}^{d} o_i \cdot \log \frac{o_i}{w_i}. \tag{12}$$

3.2.5 Final Training Objective. While pre-training the Webformer, we also need to maintain the ability of context understanding of the Text Encoder. Therefore, we still add MLM loss to the Text Encoder and denote it as L_{MLM}. The implementation process is the same as BERT [11]. We add these five parts of the loss together as the overall loss of the model:

$$L = L_{\mathrm{mnp}} + L_{\mathrm{pcm}} + L_{\mathrm{snm}} + L_{\mathrm{cop}} + L_{\mathrm{mlm}}. \tag{13}$$

3.3 Fine-tuning stage of downstream tasks

In this stage, we evaluate the effectiveness of our approach Webformer in downstream document ranking task.

We follow previous approaches of utilizing pre-trained models for ad-hoc document ranking [24, 25, 27]. Specifically, for a query q and a candidate document d, we firstly add special tokens and concatenate them as $Y = ([\mathrm{CLS}]; q; [\mathrm{SEP}]; d; \mathrm{SEP}])$, where [;] is the concatenation operation. [CLS] token is used to summarize the whole sequence for special needs of downstream tasks. [SEP] token is used to mark the end of the query and document. Then, we feed the sequence Y to Webformer, and use the representation of [CLS] to get the matching score:

$$\mathrm{s(q, d)} = \mathrm{MLP(h_{cls})}, \mathrm{h_{cls}} = \mathrm{CLS(Webformer(Y))}. \tag{14}$$

We use cross-entropy loss as the objective:

$$L_{\mathrm{dr}} = \frac{1}{N} \sum_{i=1}^{N} y_i \log \mathrm{s(q, d)} + (1 - y_i) \log(1 - \mathrm{s(q, d)}), \tag{15}$$

where N is the number of samples in training set.

4 EXPERIMENTS

4.1 Datasets and Evaluation Metrics

4.1.1 Pre-training Corpus. In the pre-training phase of Webformer, we use WikiExtractor[2] to extract the data in Wikidump[3] and keep the HTML tags. To guarantee enough structural information and remove noise data, we do the following data cleaning: (1) We remove self-closing tags and script tags. We remove the self-closing tags which are used for standardizing the page structure and do not contain text information, such as "<meta>" and "<input>" tags. (2) We remove the page comments which are not related to the page content, such as the texts between "<!—" and "—>" in HTML text. (3) Following [5, 21], we remove the internal attributes of the tags. We treat a tag as a single token, thus we standardize all tags into the format of "<tag_name>". (4) We remove the page whose total length is less than 512, since it can be directly put into a BERT structure. Finally, we obtain 2,975,354 web pages with HTML sources for pre-training.

Note that we use Wikipedia as our pre-training corpus, because it is widely used in previous work [7, 11, 16, 28, 40]and makes our work comparable to existing methods. Pre-training with a larger corpus, such as CommonCrawl [4] or WebText [30], is definitively valuable to industry, but we do not have enough computation resources. We are looking forward to collaborating with our partners on this task in the future.

4.1.2 Fine-tuning Datasets. To evaluate our proposed model Webformer, we conduct fine-tuning experiments on two authoritative ad-hoc retrieval datasets.

- **MS MARCO Document Ranking (MS MARCO)**[5] [23]: MS MARCO is a large-scale benchmark dataset on document ranking. It consists of 3.2 million documents with 367 thousand training queries, 5 thousand development queries, and 5 thousand test queries. The relevance is measured in 0/1.
- **TREC 2019 Deep Learning Track (TREC DL)**[6] [8]: Its training set is the same as MS MARCO, but use a novel test set which has more comprehensive notations. It has 43 test queries, and the relevance is scored in 0/1/2/3.

4.1.3 Evaluation Metrics. We use MRR@100 and MRR@10 to measure the top-ranking performance in MS MARCO, and use nDCG@10 and nDCG@100 to measure the ranking performance in TREC-DL.

4.2 Baselines

We evaluate the performance of our approach by comparing it with three groups of highly related and strong baseline models:

(1) *Traditional models.* **QL** [43] is based on the Dirichlet language model and is one of the best retrieval algorithms. **BM25** [31] is another excellent retrieval algorithm. It generates the morpheme of the query, calculates and sums the correlation score of each morpheme with the candidate document to get the final score.

(2)*Neural models.* **DRMM** [13] is a typical interactive model. It first calculates the similarity between each query word and the document, counts a similarity histogram for each word, and then sends the similarity histogram for each word to a neural network to get a score, and all the scores are finally added together as the final score. **DUET** [22] is a hybrid method that combines signals from local models for correlation matching and distributed models for semantic matching. **KNRM** [37] uses kernel pooling functions to get the matching score of the query and corresponding documents. **Conv-KNRM** [37] adds a convolutional layer for modeling n-gram soft matches and fuse the contextual information of surrounding words for matching.

(3) *Pre-trained Models.* **BERT** [11] is a bi-directional Transformer pre-trained with Masked Language Modeling and Next Sentence Prediction tasks. **ICT** [6] is a pre-training task which predicts whether a batch of sentences is the context of the given query. **PROP** [18] use Representative Words Prediction task to learn the correlations between sampled word sets, and achieve the state-of-the-art performance. PROP_{Wiki} and PROP_{MARCO} represent the PROP model trained on Wikipedia and MS MARCO, respectively

4.3 Implementation details

4.3.1 Model Architecture. In our model Webformer[7], we have two Transformers, namely Text Encoder and Node Encoder. For Text Encoder, in order to facilitate the comparison with BERT[11], we use the same Transformer encoder architecture as $\mathrm{BERT-Base}$. The hidden size is 768, the number of transformer layers is 12, and the number of self-attention heads is 12. Besides, since Text Encoder needs to encode HTML text, and the semantic space of the tag tokens contained in it is different from that of the plain text, we firstly add all the tag tokens to the vocabulary, and add a tag embedding on the basis of Transformer's original embeddings. Specifically, we mark text token, start tag token and close tag token as 0/1/2 separately. For Node Encoder, we use the same configuration as above except that the number of Transformer layers is 1. We use the HuggingFace's Transformers for the model implementation [36].

4.3.2 Pre-training Stage. Due to the limitation of computing resources, we have adopted a variety of schemes to estimate the size of the model and the cost of calculation:

(1) Set a maximum layer depth k. After the page compression, the first k-1 layers of a DOM tree are kept as they are, and the parts below the k layer are merged into the bottom k layer. We counted the proportion that the sequence length of leaf nodes is less than 512, which is the maximum sequence length that BERT can encode. When the number of layers is 4, the ratio is 81.3%, when it is 5, it reaches 98.6%, and when it is 6, it reaches 99.5%. Therefore, for the consideration of efficiency and information integrity, the number of layers was set to 5.

(2) Share encoder parameters. Since each layer of the same type of Encoder has the same function, we shared the parameters of all Text Encoders in 5 layers and shared the parameters of all Node Encoders in 5 layers.

(3) Set a maximum length. The number of children of a non-leaf node is limited, most non-leaf nodes only have single digits. To indicate it, We counted the number of child nodes under each

[2]https://github.com/attardi/wikiextractor
[3]https://dumps.wikimedia.org/enwiki/
[4]https://commoncrawl.org/
[5]https://github.com/microsoft/MSMARCO-Document-Ranking
[6]https://microsoft.github.io/msmarco/TREC-Deep-Learning-2019.html
[7]https://github.com/xrr233/Webformer

Table 1: Evaluation results on MS MARCO and TREC-DL. "*" denotes the result is significantly worse than our method Webformer in t-test with $p < 0.05$ level. The best results are in bold.

Model Type	Model	MS MARCO				TREC-DL 2019			
		ANCE		OFFCIAL		ANCE		OFFCIAL	
		MRR@100	MRR@10	MRR@100	MRR@10	ndcg@100	ndcg@10	ndcg@100	ndcg@10
Traditional Models	QL	.2457*	.2295*	.2103*	.1977*	.4644*	.5370*	.4694*	.4354*
	BM25	.2538*	.2383*	.2379*	.2260*	.4692*	.5411*	.4819*	.4681*
Neural Models	DRMM	.1146*	.0943*	.1211*	.1047*	.3812*	.3085*	.4099*	.3000*
	DUET	.2287*	.2102*	.1445*	.1278*	.3912*	.3595*	.4213*	.3432*
	KNRM	.2816*	.2740*	.2128*	.1992*	.4671*	.5491*	.4727*	.4319*
	Conv-KNRM	.3182*	.3054*	.2850*	.2744*	.4876*	.5990*	.5221*	.5899*
Pre-trained Models	BERT-Base	.4199*	.4108*	.3768*	.3709*	.4907	.6054*	.5289	.6358*
	ICT	.4196*	.4108*	.3827*	.3770*	.4943	.6143	.5300	.6386*
	PROP_{Wiki}	.4188*	.4092*	.3818*	.3759*	.4882*	.6050*	.5251*	.6224*
	PROP_{Marco}	.4253*	.4166*	.3890*	.3837*	.4894	.6166	.5242*	.6208*
	Webformer (Ours)	**.4422**	**.4340**	**.4036**	**.3984**	**.4967**	**.6200**	**.5335**	**.6479**

Table 2: Ablation study results. "*" denotes the result is significantly worse than our method Webformer in t-test with $p < 0.05$ level. The best results are in bold.

Dataset	MS MARCO			
Model Name	ANCE		OFFICIAL	
	MRR@100	MRR@10	MRR@100	MRR@10
Webformer	**.4422**	**.4340**	**.4036**	**.3984**
w/o BERT	.4345*	.4259*	.3884*	.3829*
w/o MNP	.4350*	.4267*	.3992	.3938
w/o PCM	.4342*	.4252*	.3971*	.3921*
w/o SNM	.4336*	.4250*	.3943*	.3890*
w/o COP	.4350*	.4267*	.3980	.3928

node. 78.4% of child nodes are less than 5, 96.0% are less than 10, and 98.5% are less than 20. In addition, in a node with more than 10 child nodes, the content of most of its child nodes tends to be homogenized, such as a reference list. Therefore, for the consideration of the limitation of machine resources, we set the maximum length of Node Encoder to 10. In order to ensure the amount of information, when sampling the leaf nodes, we keep the top 10 nodes with the longest inner HTML text length and keep their relative order.

For the MLM objective, we follow the settings in BERT, where we randomly use [MASK] token to drop the token in the sentence with a probability of 15%, and the masked token has an 80% chance of being replaced with [MASK], 10% probability of being replaced with a random token, and 10% probability remains unchanged. We use the Adam optimizer with a learning rate of 5e-5 for 2 epochs, where the batch size is set as 64.

4.3.3 Fine-tuning Stage. The Transformer model used in the fine-tuning stage is the Text Encoder that has been pre-trained. And we test the rerank results of our model in document ranking. The result of each data set comes from two candidate sets, namely ance and official. The former is a candidate set based on the ance [38] retrieval model, and the latter is an official candidate set. In the process of fine-tuning, we spliced the query, url and body of the document together. When the total length exceeds 512, the body content will be cropped. Besides, since the downstream task is a plain text task, we mark all tag embedding as 0. Each of our models has been fine-tuned for two epochs, and the learning rate is set to 1e-5 with a warm-up portion 0.1.

4.4 Experimental Results

The results of our experiment are reported in Table 1, we can observe that:

(1)**Our model Webformer achieves the best results in all indicators on both datasets**. On the MS MARCO dataset (ANCE), Webformer significantly outperforms previous state-of-the-art baseline $\text{PROP}_{\text{MARCO}}$ by 4.0% regarding MRR@100. On the TREC-DL dataset (ANCE), Webformer outperforms $\text{PROP}_{\text{MARCO}}$ by 1.5% regarding nDCG@100. This finding also verifies that exploring the structural information (*e.g.*parent-child relationship, parallel relationship, etc.) between web texts can bring orthogonal effective signals to the IR model.

(2) **Generally speaking, the effect of the pre-trained IR models is better than that of the neural model, and the neural model is better than the traditional IR model**. We can see that the effects of BM25 and QL are higher than those of DRMM and DUET, indicating that the traditional retrieval model is actually a powerful baseline and has always had good results on retrieval tasks. Secondly, Conv-KNRM outperforms traditional retrieval models in all indicators, which proves that using distributed vectors to represent query and document and automatically learn the relationship between them can improve the quality of the retrieval model. In addition, we can see that the pre-training models outperform the neural models and the traditional models, which

proves that pre-training on large-scale unlabeled data and fine-tuning on specific downstream tasks can also significantly improve the effect of the retrieval model.

(3) **Among the pre-training models, designing new tasks for IR rather than directly using pre-trained models have a better performance**. Compared with BERT, both ICT task and PROP have a better performance in ad-hoc retrieval, proving that even with the pre-trained model, more specific and related task design for IR can make it perform better on downstream tasks. Besides, the PROP$_{wiki}$ and PROP$_{marco}$ are the previous state-of-the-art baselines, confirming that Representative Words Prediction task have excellent effects on downstream IR tasks. Our proposed method Webformer, which designs four pre-training tasks on the basis of the DOM tree structure to help the pre-trained model understand the structural information in web pages, also get better effects for ad-hoc retrieval. Even if we do not use the whole hierarchical model but the Text Encoder to encode pure text, Webformer achieves significant improvements compared with the existing pre-trained methods.

4.5 Further Analysis

We further analyze the influence of different tasks we proposed in ablation study (Section 4.5.1), the performance under different fine-tuning queries (Section 4.5.2), the effect of extra tokens (Section 4.5.3),and the effect of parallel information through a case study (Section 4.5.4).

4.5.1 Ablation Study. To better verify the effectiveness of each module and each self-supervised task in Webformer, we conduct ablation experiments on the MS MARCO data set, and the results are shown in Table 2. First, we seek to explore the benefit brought by the pre-trained language model. Specifically, instead of initializing the model with pre-trained parameters, we train the Text Encoder from scratch. The results show that removing pre-trained language model would lead to a performance drop, proving that the semantic prior knowledge of pre-trained language model plays an important role in the model.

Second, we explore the effects of the four self-supervised tasks. We find that removing any of the four tasks would lead to a performance drop across various evaluation indicators, demonstrating that each of the four self-supervised tasks is able to capture effective semantic signals and therefore improves the model performance. Specifically, removing the SNM task leads to the largest drop, which proves that the parallel information between the texts plays a vital role in understanding the document. Removing the MNP task has the least impact on the model, we guess that this is because the previous language model built less parallel relations and more semantic aspects. Despite this, it still causes a drop of more than 1% on each metric, showing that the ability of text understanding of Node Encoder can improve the ranking ability of our model in the pre-training phase. Removing PCM and COP tasks also cause a large amount of effect loss of the model, proving that the parent-child relationship and the relative order between the text can promote the pre-training model's understanding of the entire web page.

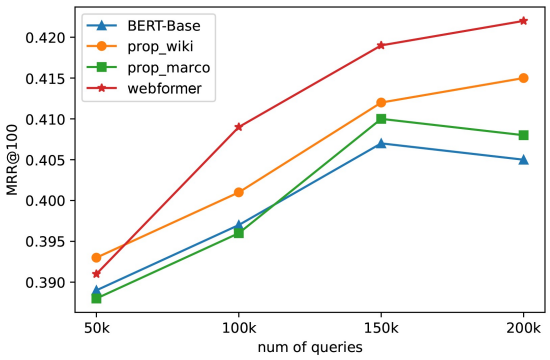

Figure 4: The effect of fine-tuning queries of Webformer on MS MARCO ANCE set.

Table 3: Evaluation results on BERT-Base and HTML-BERT, where HTML-BERT is a model that we use the same training strategy with BERT but the training data is HTML text of English Wikipedia pages. "*" denotes the result is significantly worse than our method Webformer in t-test with $p < 0.05$ level. The best results are in bold.

Dataset	MS MARCO			
Model Name	ANCE		OFFICIAL	
	MRR@100	MRR@10	MRR@100	MRR@10
Webformer	**.4422**	**.4340**	**.4036**	**.3984**
BERT-Base	.4199*	.4108*	.3768*	.3709*
HTML-BERT	.4218*	.4135*	.3849*	.3799*

4.5.2 Effect of Fine-tuning Queries. To better show the effectiveness of our pre-trained model on downstream ranking tasks, we fine-tune our model with a varying number of queries (*i.e.*, 50K,100K, 150K,200K). We report MRR@100 to evaluate the performance. Figure 4 presents the results of MS MARCO ANCE Top100. We can find that: (1) with the number of fine-tuning queries increasing, our Webformer consistently obtains incremental improvements, which implies the great model capacity and generalization ability of Webformer. (2) in 50k steps, Webformer performs worse than PROP (wiki). The potential reason might be that Webformer is pre-trained with a hierarchical model structure that requires more data to fit. After 50k steps, the performance of Webformer steadily increases and outperforms baseline models by a large margin, which indicates that utilizing structure information of web pages indeed brings orthogonal structural semantics that plain text fails to provide.

4.5.3 Effect of extra tokens. In order to better understand the role of our model structure and extra HTML tag tokens, we conduct an experiment to compare the performance of BERT-Base and HTML-BERT, where HTML-BERT is trained on the same dataset of our proposed Webformer, but use the flatten HTML text.

Specifically, for each piece of HTML text data, we first construct it into a tree structure based on HTML tags, and then perform a depth-first search on its nodes. If the length of the inner HTML text of the node is less than 512, we will use it as a piece of training

data, otherwise continue to search downwards. In this way, we can better maintain the integrity of semantic information. Besides, in order to model short texts and the connection between parent and child nodes, when we generate a piece of training data of the parent node, there is a 20% probability of extracting HTML text from its child node again as a new piece of training data. Following the above method, we traverse a DOM tree and generate multiple Bert training data. Based on this, we obtained a total of 89,232,208 training data.

We train the HTML-BERT model with the MLM objective for two epochs, the settings are the same as BERT, then apply it to downstream document ranking task on MS MARCO dataset. The results are shown in Table 3, we can see that although Webformer have the same amount of data with HTML-BERT, it outperforms the HTML-BERT model in all metrics, proving that our model structure and the pre-training tasks play an important role on the performance improvement on downstream IR tasks. Besides, HTML-BERT's performance is slightly better than BERT-Base, especially in MS MARCO official dataset, indicating that extra tokens (*i.e.*, HTML tag tokens) have an opposite effect to corresponding downstream tasks, but still have much room for improvement.

4.5.4 Effect of Parallel information. Since the parallel structure is the most easily overlooked information in traditional pre-training models, we conduct a case study to study the model's ability to understand parallel information. Detailedly, in the navigation bar at the top of the main page of English Wikipedia, there is a list of options that classify the content of all Wikipedia pages and these options are semantically parallel, including arts, biology, mathematics, *etc.*. In addition, we have added the word "apple" as an interference item.

We pass the sequence composed of these options to BERT-Base and Webformer respectively, and use attention weight to study its parallel information understanding ability. The result is shown in Figure 5, it can be seen that (1) In the results of BERT-Base, apple's attention weight still accounts for a large proportion, even if it has no connection with other information. Besides, the attention distribution of each word is similar. This shows that the model cannot capture the parallelism of other tokens, so it equates apple with other tokens. (2) In our model, except for itself, apple's attention weight has very low weights, which proves that the model believes that apple and other tokens are not in the same semantic space. On top of this, the corresponding words such as "science" and "technology" has a high weight, indicating that our model has learned the semantic connection between parallel words.

5 CONCLUSION AND FUTURE WORK

In this paper, we propose a new pre-training model framework Webformer. Compared with existing work, we incorporate the rich structural information in the HTML text data of web pages into the pre-trained model. We first build a hierarchical model architecture by imitating the structure of the DOM tree, and design two kinds of Transformers to encoder leaf nodes and non-leaf nodes separately. Then, we initialize the Text Encoder using a pre-trained model. Based on this architecture, we design four pre-training tasks to help the model capture the structural information in web pages in different views. In the fine-tuning stage, we use the pre-trained

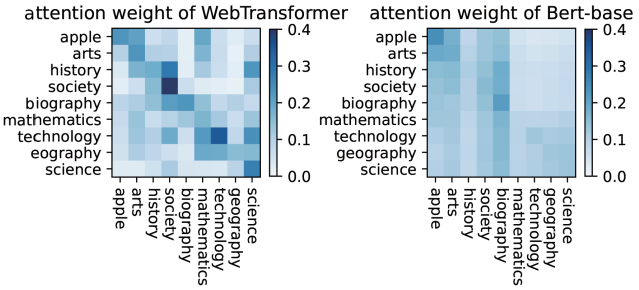

Figure 5: Webformer's and BERT's attention weights of the sequence composed of parallel words

model to test the effect of the downstream document ranking task. The results in MS MARCO and TREC-DL 2019 prove the effectiveness of our model.

However, the pre-training HTML dataset we use is well-formed Wikipedia, and we will continue our work on more general HTML datasets such as ClueWeb09[8]. Besides, due to the lack of authoritative HTML-formatted ad-hoc retrieval datasets, we only use Text Encoder for downstream tasks in this paper, and we will explore whether the tags can also be used to learn better web page representations when the corresponding HTML source code is also available.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China No. 61872370, Beijing Outstanding Young Scientist Program NO. BJJWZYJH012019100020098, China Unicom Innovation Ecological Cooperation Plan, Beijing Academy of Artificial Intelligence(BAAI), and Intelligent Social Governance Platform, Major Innovation & Planning Interdisciplinary Platform for the "Double-First Class" Initiative, Renmin University of China. The work was partially done at Key Laboratory of Data Engineering and Knowledge Engineering, MOE. We also acknowledge the support provided and contribution made by the Public Policy and Decision-making Research Lab of RUC.

REFERENCES

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

[2] Ammar Al-Dallal and Rasha S Abdul-Wahab. 2011. Achieving high recall and precision with HTLM documents: an innovation approach in information retrieval. In *Proceedings of the World Congress on Engineering*, Vol. 3.

[3] Colin Ashby and David Weir. 2020. Leveraging HTML in Free Text Web Named Entity Recognition. In *Proceedings of the 28th International Conference on Computational Linguistics*. 407–413.

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

[5] Sebastian Blohm. 2011. *Large-scale pattern-based information extraction from the world wide web.* KIT Scientific Publishing.

[6] Wei-Cheng Chang, Felix X. Yu, Yin-Wen Chang, Yiming Yang, and Sanjiv Kumar. 2020. Pre-training Tasks for Embedding-based Large-scale Retrieval. In *ICLR 2020*. OpenReview.net.

[8] https://lemurproject.org/clueweb09/

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

[8] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. 2020. Overview of the TREC 2019 deep learning track. (2020).

[9] Michal Cutler, Yungming Shih, and Weiyi Meng. 1997. Using the Structure of HTML Documents to Improve Retrieval.. In *USENIX Symposium on Internet Technologies and Systems*. 241–252.

[10] Zhuyun Dai and Jamie Callan. 2019. Deeper Text Understanding for IR with Contextual Neural Language Modeling. In *SIGIR 2019*. ACM, 985–988.

[11] 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 NAACL 2019*. ACL, 4171–4186.

[12] Luyu Gao, Zhuyun Dai, and Jamie Callan. 2021. Rethink Training of BERT Rerankers in Multi-stage Retrieval Pipeline. In *ECIR 2021*. Springer, 280–286.

[13] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. 2016. A Deep Relevance Matching Model for Ad-hoc Retrieval. In *CIKM 2016*. ACM, 55–64.

[14] Sun Kim and Byoung-Tak Zhang. 2003. Genetic mining of HTML structures for effective web-document retrieval. *Applied Intelligence* 18, 3 (2003), 243–256.

[15] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The Efficient Transformer. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. https://openreview.net/forum?id=rkgNKkHtvB

[16] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. https://openreview.net/forum?id=H1eA7AEtvS

[17] Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent Retrieval for Weakly Supervised Open Domain Question Answering. In *Proceedings of the ACL 2019*. ACL, 6086–6096.

[18] Xinyu Ma, Jiafeng Guo, Ruqing Zhang, Yixing Fan, Xiang Ji, and Xueqi Cheng. 2021. PROP: Pre-Training with Representative Words Prediction for Ad-Hoc Retrieval. In *Proceedings of the WSDM 2021 (WSDM '21)*. ACM, 283–291.

[19] Xinyu Ma, Jiafeng Guo, Ruqing Zhang, Yixing Fan, Yingyan Li, and Xueqi Cheng. 2021. B-PROP: Bootstrapped Pre-training with Representative Words Prediction for Ad-hoc Retrieval. In *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, Fernando Diaz, Chirag Shah, Torsten Suel, Pablo Castells, Rosie Jones, and Tetsuya Sakai (Eds.). ACM, 1318–1327. https://doi.org/10.1145/3404835.3462869

[20] Zhengyi Ma, Zhicheng Dou, Wei Xu, Xinyu Zhang, Hao Jiang, Zhao Cao, and Ji-Rong Wen. 2021. Pre-training for Ad-hoc Retrieval: Hyperlink is Also You Need. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 1212–1221.

[21] Marcin Michał Mirończuk. 2018. The BigGrams: the semi-supervised information extraction system from HTML: an improvement in the wrapper induction. *Knowledge and Information Systems* 54, 3 (2018), 711–776.

[22] Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to Match using Local and Distributed Representations of Text for Web Search. In *WWW 2017*.

[23] Tri Nguyen, Mir Rosenberg, Xia Song, et al. 2016. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. In *NIPS 2016*.

[24] Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. *CoRR* abs/1901.04085 (2019).

[25] Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019. Multi-Stage Document Ranking with BERT. *CoRR* abs/1910.14424 (2019).

[26] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of the NAACL 2018*. ACL, 2227–2237.

[27] Yifan Qiao, Chenyan Xiong, Zhenghao Liu, and Zhiyuan Liu. 2019. Understanding the Behaviors of BERT in Ranking. *CoRR* abs/1904.07531 (2019). arXiv:1904.07531 http://arxiv.org/abs/1904.07531

[28] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. In *Proceedings of Techinical re-port, OpenAI*.

[29] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2018. Language Models are Unsupervised Multitask Learners. (2018).

[30] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[31] Stephen E. Robertson and Steve Walker. 1994. Some Simple Effective Approximations to the 2-Poisson Model for Probabilistic Weighted Retrieval. In *Proceedings of the SIGIR 1994*. ACM/Springer, 232–241.

[32] Zhan Su, Zhicheng Dou, Yutao Zhu, Xubo Qin, and Ji-Rong Wen. 2021. Modeling Intent Graph for Search Result Diversification. In *Proceedings of the SIGIR 2021*. ACM, 736–746. https://doi.org/10.1145/3404835.3462872

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

[34] Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-Attention with Linear Complexity. *CoRR* abs/2006.04768 (2020). arXiv:2006.04768 https://arxiv.org/abs/2006.04768

[35] Mengxi Wei, Yifan He, and Qiong Zhang. 2020. Robust Layout-aware IE for Visually Rich Documents with Pre-trained Language Models. In *SIGIR 2020*.

[36] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *CoRR* abs/1910.03771 (2019). arXiv:1910.03771 http://arxiv.org/abs/1910.03771

[37] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-End Neural Ad-hoc Ranking with Kernel Pooling. In *SIGIR 2017*. 55–64.

[38] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, et al. 2020. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. *CoRR* abs/2007.00808 (2020).

[39] Wei Yang, Haotian Zhang, and Jimmy Lin. 2019. Simple Applications of BERT for Ad Hoc Document Retrieval. *CoRR* abs/1903.10972 (2019).

[40] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems* 32 (2019).

[41] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *NIPS 2019*. 5754–5764.

[42] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big Bird: Transformers for Longer Sequences.. In *NeurIPS*.

[43] Chengxiang Zhai and John D. Lafferty. 2017. A Study of Smoothing Methods for Language Models Applied to Ad Hoc Information Retrieval. *SIGIR Forum* 51, 2 (2017), 268–276.

[44] Yujia Zhou, Zhicheng Dou, Huaying Yuan, and Zhengyi Ma. 2022. Socialformer: Social Network Inspired Long Document Modeling for Document Ranking. *CoRR* abs/2202.10870 (2022).

[45] Yujia Zhou, Zhicheng Dou, Yutao Zhu, and Ji-Rong Wen. 2021. PSSL: Self-supervised Learning for Personalized Search with Contrastive Sampling. In *CIKM*. ACM, 2749–2758.

[46] Yutao Zhu, Jian-Yun Nie, Zhicheng Dou, Zhengyi Ma, Xinyu Zhang, Pan Du, Xiaochen Zuo, and Hao Jiang. 2021. Contrastive Learning of User Behavior Sequence for Context-Aware Document Ranking. In *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021*, Gianluca Demartini, Guido Zuccon, J. Shane Culpepper, Zi Huang, and Hanghang Tong (Eds.). ACM, 2780–2791. https://doi.org/10.1145/3459637.3482243

[47] Yutao Zhu, Kun Zhou, Jian-Yun Nie, Shengchao Liu, and Zhicheng Dou. 2021. Neural Sentence Ordering Based on Constraint Graphs. In *AAAI 2021*. AAAI Press, 14656–14664. https://ojs.aaai.org/index.php/AAAI/article/view/17722