WebSRC: A Dataset for Web-Based Structural Reading Comprehension

Xingyu Chen, Zihan Zhao, Lu Chen,*
Jiabao JI, Danyang Zhang, Ao Luo, Yuxuan Xiong and Kai Yu*
X-LANCE Lab, Department of Computer Science and Engineering
MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University
Shanghai Jiao Tong University, Shanghai, China
State Key Lab of Media Convergence Production Technology and Systems, Beijing, China
{galaxychen, zhao_mengxin, chenlusz}@sjtu.edu.cn,
{sjcs_jijiabao, zhang-dy20, wenzelmetternich, xiongyx, kai.yu}@sjtu.edu.cn

Abstract

Web search is an essential way for humans to obtain information, but it's still a great challenge for machines to understand the contents of web pages. In this paper, we introduce the task of structural reading comprehension (SRC) on web. Given a web page and a question about it, the task is to find the answer from the web page. This task requires a system not only to understand the semantics of texts but also the structure of the web page. Moreover, we proposed WebSRC, a novel **Web**-based **S**tructural **R**eading **C**omprehension dataset. WebSRC consists of 400K question-answer pairs, which are collected from 6.4K web pages. Along with the QA pairs, corresponding HTML source code, screenshots, and metadata are also provided in our dataset. Each question in WebSRC requires a certain structural understanding of a web page to answer, and the answer is either a text span on the web page or yes/no. We evaluate various baselines on our dataset to show the difficulty of our task. We also investigate the usefulness of structural information and visual features. Our dataset and baselines have been publicly available[1].

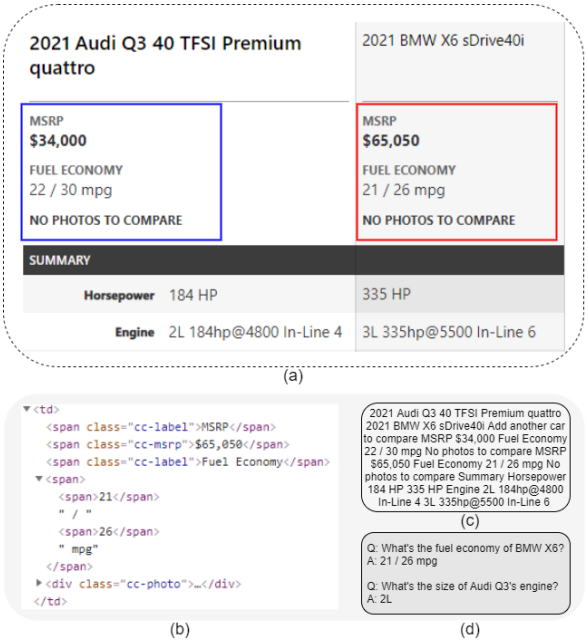


Figure 1: Examples for a web page. (a) is the original web page. (b) is the HTML code for the content in the red box. Each HTML tag begins with a starting tag and ends with a closing tag (with a slash in tag). *<td>* stands for a table cell and ** stands for a content span. (c) shows the text extracted from the web page. (d) contains some sample questions for the web page.

1 Introduction

Web pages are the most common source of human knowledge and daily information. With the help of modern search engines, people can easily locate web pages and find information by simply typing some keywords. However, traditional search engines only retrieve web pages related to the query and highlight the possible answers (Chen, 2018), they can't understand the web pages and answer the query based on contents. The rapid development of question answering systems and knowledge graphs enables search engines to answer simple questions directly (Chakraborty et al., 2019), but they still fail to perform question answering on arbitrary web pages. The difficulty lies in the variety of web pages and the complexity of the web layouts, which requires a system not only to consider the text but also the structures of web pages.

There are two kinds of structures for each web page: *spatial* structure and *logical* structure. The spatial structure is how the information is visually organized, and the logical structure is how the information is organized by semantics. Figure 1(a) shows the spatial structure of the web page, e.g., how the texts are arranged and what are their relative positions. The logical structure can be deduced by the spatial structure and the semantics of the texts. For example, this image introduces the infor-

*The corresponding authors are Lu Chen and Kai Yu.
[1]https://x-lance.github.io/WebSRC/

Datasets	#domain	#website	Task	#Query	With Image
WEIR(Bronzi et al., 2013)	4	40	ClosedIE	32	No
SWDE(Hao et al., 2011)	8	80	ClosedIE	32	No
Expanded SWDE(Lockard et al., 2019)	3	21	OpenIE	748	No
WebSRC(Ours)	11	70	QA	2735	Yes

Table 1: The comparison with datasets with HTML. The query in ClosedIE is the attributes needed to be extracted, in OpenIE is predicates, and in WebSRC is the questions before data augmentation.

mation about two cars, with car names at the top followed by the detailed specifications. A human can easily answer the questions in Figure 1(d) by referring to the relevant section in the logical structure. But for computers, it's hard to understand the logical structure by just taking the spatial structure (the image) as input due to the lack of common sense. Computers need to infer the answers from the font size, the color and the spatial relations between texts, let alone they need to extract texts from the image and understand them.

An alternative way is to utilize the text from web page. Figure 1(c) shows the texts extracted from Figure 1(a). As we can see, the layout structure is lost in the plain text, and the text is just a concatenation of short phrases without a meaningful context. It would be difficult to answer questions only based on such texts. Besides texts, we can also parse the HTML (Hypertext Markup Language) document, i.e. the source code of the web page. It describes the structure of the webs page and uses *HTML elements (tags)* to display the contents. We will use the term *tag* and *element* in this paper interchangeably. Figure 1(b) shows the HTML code corresponding to the part of the web page highlighted in the red box. HTML is a kind of semi-structured document (Buneman, 1997), where tags with different structural semantics serve as separators. It's also called the "self-describing" structure. An HTML document can be parsed into a tree-like structure called DOM[2] (Document Object Model), where the tree nodes are elements in the HTML, and texts are all leaf nodes in the tree. An HTML DOM tree can serve as a structural representation of the web page, where visually similar items on the web page would be sub-trees with similar structures. For example in Figure 1, the HTML structure is identical for the segment in the blue and red boxes. They are only different in the text. However, due to the complexity of rendering HTML code into a web page, a single HTML would not be enough to represent the full logic structure of the web page. For example, in Figure 1(b), the four <*span*> are in the same spatial level of the DOM tree, but they play different semantic roles in the web page, i.e. the first span indicates an attribute and the second contains the corresponding value. We need to leverage both the visual and structural information to gain a comprehensive understanding.

To promote researches in question answering on web pages, we introduce WebSRC, a dataset for reading comprehension on structural web pages. The task is to answer questions about web pages, which requires a system to have a comprehensive understanding of the spatial structure and logical structure. WebSRC consists of 6.4K web pages and 400K question-answer pairs about web pages. For each web page, we manually chose one segment from it and saved the corresponding HTML code, screenshot, and metadata like positions and sizes. Questions in WebSRC were created for each segment. Answers are either text spans from web pages or yes/no. Taking the HTML code, screenshot, metadata as well as question as input, a model is to predict the answer from the web page. The comparison of WebSRC with other datasets with HTML documents is illustrated in Table 1. Our dataset is the only one that provides HTML documents and images, and is larger in the number of domains and queries.

To summarize, our contributions are as follows:

- We proposed the task of structural reading comprehension (SRC) on web, which is a multi-modal machine reading comprehension task that focuses on understanding texts and screenshots on web pages.
- We created a large dataset for web-based structural reading comprehension consisting of 400K QAs and 6.4K web page segments, where HTML code and additional visual features are also provided.
- We evaluated several baselines on WebSRC

[2]https://en.wikipedia.org/wiki/Document_Object_Model

and the results showed that WebSRC is highly different from the existing textual QA datasets and is challenging even for the leading pre-trained language model.

2 Related Work

Machine reading comprehension (MRC) models have achieved excellent performance on plain text corpus (Zeng et al., 2020) in recent years. Traditional datasets for machine reading comprehension (Talmor et al., 2019; Yang et al., 2018; Rajpurkar et al., 2016, 2018; Choi et al., 2018; Reddy et al., 2019; Lai et al., 2017) contain plain text passages and QAs about them. However, HTML code in the form of semi-structured documents is different from the ordinary textual corpus. Recently, multi-modal MRC has gained the interest of researchers. Multi-modal MRC datasets with both images and texts are proposed, such as MovieQA (Tapaswi et al., 2016), TQA (Kembhavi et al., 2017), COMICS (Iyyer et al., 2017) and RecipeQA (Yagcioglu et al., 2018). Images in these datasets provide different information from texts, and texts are supplementary descriptions for images. Text VQA (Mishra et al., 2019; Singh et al., 2019; Mathew et al., 2021) is a kind of VQA (visual question answering) task (Antol et al., 2015), whose task is to answer questions about a real-world image, and questions in this task are about the texts in the image. However, there is no existing text or layout description available in the image, but we can access them easily on web pages.

Information extraction for web pages has been investigated intensively (Chang et al., 2006). Previous studies mainly focus on building templates for HTML DOM tree, called Wrapper Induction (Kushmerick, 2000; Flesca et al., 2004; Kushmerick et al., 1997; Muslea et al., 1999), or using well designed visual features like font sizes, element sizes, and positions (Zhu et al., 2005, 2006). These methods require abundant human labor to label templates and analyze features, which makes it hard to generalize to unseen websites. Chen et al. (2021) proposed a program synthesis based technique to extract web information. Some studies focused on recognizing tables from web pages (Zanibbi et al., 2004) and tried to model the physical and logical structure of tables in HTML, Zhang et al. (2020) proposed to use a graph to represent the table structure. Some web QA datasets are proposed (Dunn et al., 2017; Joshi et al., 2017; Dhingra et al., 2017; Li et al., 2016), but they only contain text snippets extracted from web pages. Bronzi et al. (2013) proposed a dataset called WEIR, consisting of 40 websites from 4 domains. Hao et al. (2011) proposed SWDE, which contains 124,291 web pages from 80 websites, and Lockard et al. (2019) expanded SWDE for openIE. All these datasets only contain HTML code for extraction, and the task is to extract pre-defined attributes of entities in web pages, e.g. the author of a book. Layout analysis (Binmakhashen and Mahmoud, 2019) is the task to analyze document images like contracts, bills, and business emails. IIT-CDIP (Lewis et al., 2006) and RVL-CDIP (Harley et al., 2015) are two datasets collected for document classification. Jaume et al. (2019) proposed FUNSD for form understanding and Huang et al. (2019) organized SROIE competition for receipt understanding. PubLayNet (Zhong et al., 2019) and DocBank (Li et al., 2020) are proposed to benchmark the task of layout recognition in academic papers. However, compared to the images in the layout analysis task, web pages are much more complex in organizing information. The terms to be recognized are relatively stable in layout analysis, while web pages may contain various information that is hard to be pre-defined.

3 Data Collection

The construction of our dataset consists of five stages: § 3.3 web page selection, § 3.4 web page collection, § 3.5 question labeling, § 3.6 data augmentation and § 3.7 final review. We will describe each stage in detail below.

3.1 Task Definition

The task of structural machine reading comprehension on web can be described as given the context $\mathcal{C}$ and a question q, predict the answer a. In our task, the context can be HTML code, screenshots, and the corresponding metadata. Denote the machine reading comprehension model as $\mathcal{F}$, our task can be formulated as:

$$\mathcal{F}(\mathcal{C}, q) = a \tag{1}$$

3.2 Locating text in HTML

To precisely locate the text in HTML, we first define **the text of an HTML node**: the text of an HTML node is the concatenation of texts in its descendant nodes in the DOM tree, where the order of texts is derived by the depth-first search. With

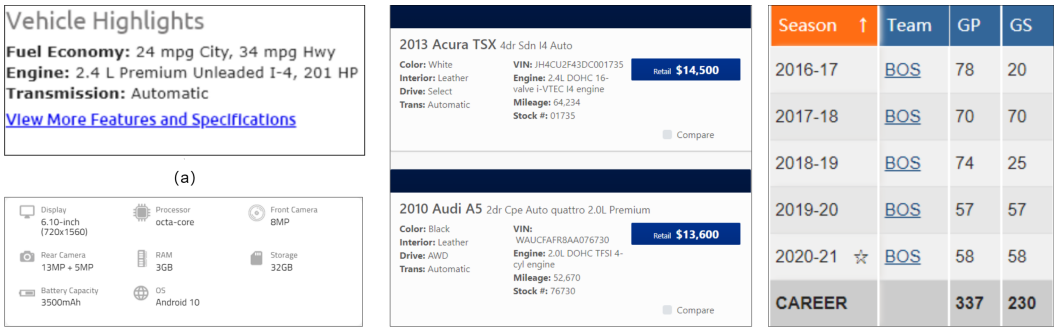


Figure 2: Examples for three types of web pages. (a) and (b) are web pages of type *KV*, (c) is a web page of type *comparison*, (d) is a web page of type *table*.

this definition, a text can be located by the tag containing the text, and the beginning position in *the text of the node*.

3.3 Web page selection

In this phase, we choose websites for further data collecting. We are interested in the structure of the web page, so in the web page selection phase, we only focused on websites with a relatively complex structure and that have abundant information for question answering. We didn't choose websites with long textual paragraphs like Wikipedia, where the structure has little influence on understanding the content. We started from the website list of the SWDE (Hao et al., 2011) dataset, which contains 80 websites from 8 domains. Websites on the list that are no longer available are dropped. We also expanded our website list by searching the domain keywords and selected the most relevant websites. In total, we obtained 70 websites from 11 domains.

We didn't use the whole web page but only chose some segments to build our dataset, because a complete web page may contain ads or additional structures like navigation tabs, which brings too much noise into the web page and makes the task much harder. We admit that in the real-world scenario we have to deal with the full web page, but we consider the problem of question-answering in full web pages can be modeled as a two-stage process: first, find the relevant segment in the web page and then answer the question based on the segment. In this work, we will focus on learning the structure of a given web page segment and leave the segment locating problem as future work.

The choice of the segment is based on the type of web page. We category web pages into three types, *KV*, *comparison*, and *table*, according to the different ways to display information. We will discuss different types of websites in detail below.

KV Information in this type of web page is presented in the form of "*key: value*", where the *key* is an attribute name and the *value* is the corresponding value. See Figure 2(a) and Figure 2(b) for illustration. This kind of web page can be found from the detail page of an entity, e.g. a car or a book. We choose the section that describes attributes about the entity from the web page.

Comparison This type is similar to type KV but with a major difference: web pages of type comparison contain several entities with the same attributes. For instance, in Figure 2(c), there are two cars with same attributes in the image and they form a comparison. We chose the segment that at least contains a comparison between two objects.

Table Web pages of this type use a table to present information. A table contains the comparison between rows naturally but unlike the type comparison, it uses a unified header to represent attributes and each row in the table only contains values. Figure 2(d) shows the statistics table of a basketball player. The segment we chose is the table area on the web page.

3.4 Web page collection

We recruited six computer science students with web crawling experience to collect the web pages. We first rendered the website in the headless Chrome browser, then for each segment, we manually wrote extracting code to crawl it. We saved the corresponding HTML and the screenshot of the segment, as well as additional metadata (including the location and size of each tag, the color and font of texts). We used Selenium[3] to collect all the data.

For segments of type comparison or type table, we would drop some objects in comparison

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

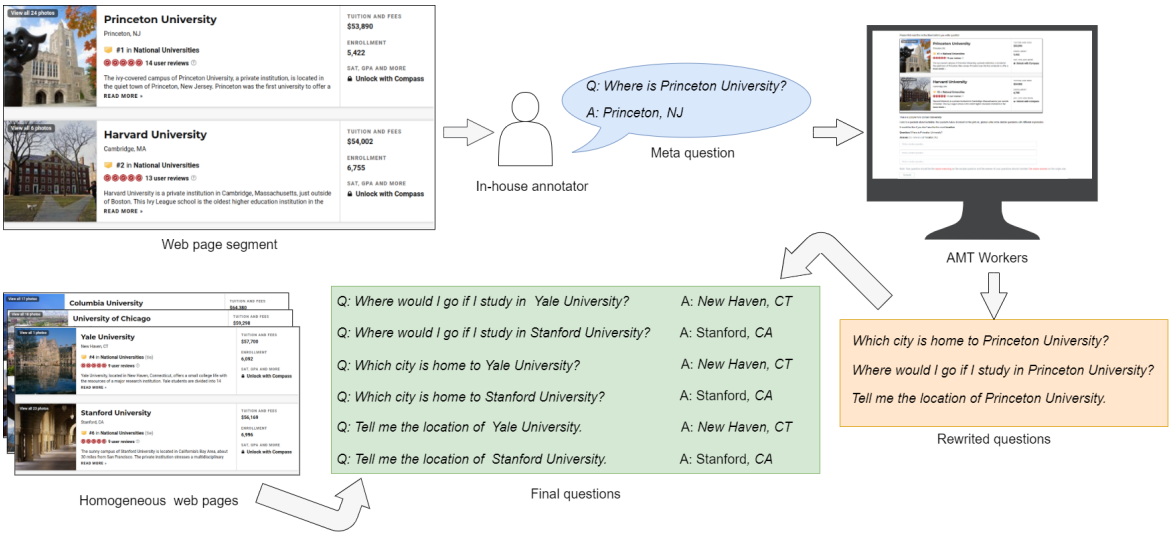


Figure 3: The pipeline for labeling questions and augmenting data.

or delete some rows in the table if the size of the segment is too large. We crawled homogeneous segments from 100 web pages under each website, each of them shares a common HTML structure but with different content. We obtained 6447 web pages after dropping some invalid web pages. We removed all non-ascii characters and extra spaces in HTML. Tags that have little influence on HTML structure are removed, including the <*script*> and the <*style*>. Properties of HTML tags are also removed except for *class*, *id*, *title* and *aria-label*, for these properties often serve as descriptions of a tag. We added an additional attribute called *tid* to all tags, which was used in locating tags with answers.

3.5 Question Labeling

We recruited three annotators to label questions and answers for each crawled segment. We showed screenshots to annotators and asked them to create questions about the content on the image. All questions should be answerable by the screenshot, and the answer should be a text shown in the image or yes/no. We asked annotators to create questions in the following style:

- Ask questions about certain key-value pair. For example in Figure 2 (a), *what's the engine specification of this car?*
- Ask questions about certain object in the comparison. For example in Figure 2 (c), *what's the price of Audi A5?*
- Ask questions about a cell value in the table. For the table example in Figure 2 (d), *what's the GP score in 2017-18?*
- Ask questions with condition. For example in Figure 2 (c), *what's the price of the white car?* with a condition "white".
- Ask yes/no questions for confusing terms. For example in Figure 2 (b), *Is the storage 32GB?* asks about the storage size which is similar to the RAM.

We also asked annotators to label the answer in the HTML, including the answer text, the tag containing the answer (represented using *tid*) and the beginning position of the answer in *the text of the answer tag*. When creating questions, we also encourage annotators to ask questions that are meaningful from an actual end user's perspective.

We asked a different annotator to check if the question is followed one of the styles above and if the answer is a valid text in the segment or yes/no. We collected 460 unique questions for all segments, and we called these questions *meta-question*s.

To enhance the diversity of question expression, we published a question rewriting task on Amazon Mechanical Turk (AMT) to polish meta-questions. Workers on AMT were shown a screenshot and a meta-question with the answer, and their task is to rewrite the given question without changing the meaning. We encouraged the worker to use more complex expressions and use synonyms for attributes if possible. Each worker should create three different versions of meta-questions. 191 workers participated in the rewriting task, and

we asked another four annotators to filter questions with obvious grammar errors and inconsistent meaning. About 10% questions are dropped after review. Examples of rewritten questions are shown in Figure 3. As we can see, annotators may change the way of asking, introduce subjunctive mood or change the question to an imperative sentence. We collected 2735 questions at this stage.

3.6 Data Augmentation

Although the structure of different websites varies a lot, web pages under the same website have a similar structure. In this phase, we automatically applied collected questions to all homogeneous web pages. For each question, we manually created extracting rules to identify answers on different web pages. We generate new QA pairs for all web pages by replacing the original answer with the answer extracted from the homogeneous web page. If a question contains a specific entity name, e.g. a car name in the comparison, we also replace it with the actual entity on the corresponding web page.

After data augmentation, we obtained 400498 question-answer pairs in total. The whole process of question labeling and data augmentation is illustrated in Figure 3.

3.7 Final review

We wrote tests for the dataset to check the correctness of the label, the completeness of saved files and the format of dataset. We also sample 100 QA pairs from each website and asked four experienced annotators to double-check the correctness of semantics, e.g. whether the answer matches the question, whether the question is suitable for the web page. Cases with errors would send back to annotators for a new round of labeling.

4 Dataset Analysis

In this section, we conduct throughout analysis of WebSRC. We only show some major results here and for more statistics please refer to Appendix A.

4.1 Dataset statistics

Type	#website	#webpage	#QA
KV	34	3207	168606
Comparison	15	1339	68578
Table	21	1901	163314

Table 2: Statistics of different types of websites.

The statistics of different types of websites are shown in Table 2. The most common type of website is type KV, which accounts for about a half. The least type of website is type comparison with only 17% of the total websites. For we can generate questions for each value in a table, the proportion of QA pairs of type table is much bigger than its proportion of websites, which is about 40%.

4.2 QAs in WebSRC

WebSRC consists of two kinds of questions: wh-questions and yes-no questions. Questions starting with "what" are the most common questions, and questions starting with "what is the" account for 29.3% of the whole dataset. As for yes-no questions, words like *Is*, *Can* and *Does* are strong indicators. The average length of questions is 8.26.

Answers in WebSRC are relatively short, 86.78% of which are within 3 words and 55.21% answers have only one word. However, a text that is visually a whole may be scattered in multiple HTML tags. The example shown in Figure 1 illustrates this phenomenon. The line "*21 / 26 mpg*" is separated by "<*span*>" tags. Besides, a tag may contain additional texts except for the answer. For example, the answer to the second question in Figure 1 is a sub-span of whole tag text *2L 184hp@4800 In-Line 4*. About 2.35% answers are distributed in multiple tags and 13.21% answers are sub-spans of the text of HTML nodes.

5 Baseline Models

We propose three baseline models for WebSRC. They take different kinds of context into consideration. We describe these models in detail below.

5.1 Pre-trained Language Model with Text (T-PLM)

In the first baseline, we convert the HTML code into non-structural pure text by simply deleting all HTML tags, and utilize Pre-trained Language Models (PLM), e.g. BERT (Devlin et al., 2019), to predict answer spans. We regard it as an extractive QA task. We add two additional words *yes* and *no* to the end of context for yes-no questions prediction. Here the context $\mathcal{C}$ in Eq. (1) is the resulting plain text. The resulting plain text and the corresponding question are concatenated to form the input sequence $\mathbf{x}$. Then the probability distributions for each token to be the start token and the end token of the answer span can be obtained as

follows:

$$\mathbf{Z} = \text{PLM}\left(\mathbf{x}\right), \tag{2}$$

$$\mathbf{p}^s, \mathbf{p}^e \propto \text{SoftMax}\left(\text{Linear}\left(\mathbf{Z}\right)\right), \tag{3}$$

where $\mathbf{Z}$ is the resulting sequence representation calculated by PLM; $\mathbf{p}^s$ and $\mathbf{p}^e$ are start and end distributions. We use cross-entropy as our objective function.

In addition, after obtaining the predicted answer spans, we go over the HTML code again to find the tightest tag that contains the whole answer and take it as the predicted answer tag.

5.2 Pre-trained Language Model with HTML (H-PLM)

In the second baseline, we incorporate HTML tags into PLM. We called this baseline H-PLM. The model architecture of H-PLM is identical to T-PLM, the only difference is we use HTML documents with HTML tags as our context $\mathcal{C}$. To deal with the HTML tags, we remove all attributes, leaving the angle brackets, tag names, and the possible slashes unchanged. The resulting tag sequence looks like * , <img/>, </p>*, etc. We treat these HTML tags as new special tokens in the sequence and randomly initialize their embedding for training.

5.3 Visual Information Enhanced Pre-trained Language Model (V-PLM)

As introduced in Section 1, HTML is not enough to represent the whole web structure. In the third baseline, we take the visual information from web pages into consideration. We call this model V-PLM. It consists of three parts: PLM, visual information enhanced self-attention blocks, and a classification layer.

For each tag in HTML, we can use the bounding box provided in meta data to locate the tag in screenshot and obtain the visual embedding using the Faster R-CNN (Ren et al., 2016). We concatenate output hidden state $\mathbf{Z}$ from H-PLM with the corresponding visual embeddings, where tokens within the same tag share the same visual embedding. For the example in Figure 1, the visual embeddings of *, Fuel, Economy, * are all the same. For other special tokens and tokens in the question, their visual embeddings are zero vectors.

The concatenated embedding is then fed into a self-attention block (Vaswani et al., 2017), which is repeated N times. We repeat the concatenation procedure between each self-attention block. The final representation is then sent to the classification layer to produce the starting and ending probability distribution, which is the same as H-PLM.

6 Experiments

6.1 Dataset Splits

We manually divide our dataset into train/dev/test sets at the website level, where the training set contains 50 websites, dev and test contain 10 websites respectively. Both the dev set and the test set have all three types of websites and have a similar distribution of website types. The detailed statistics of each set are shown in Table 3.

Split	#website	#webpage	#QA
Train	50	4549	307315
Dev	10	913	52826
Test	10	985	40357

Table 3: Statistics of dataset splits.

6.2 Evaluate Metrics

We use three kinds of metrics for evaluation.

Exact match (EM) This metric is used to evaluate whether a predicted answer is completely the same as the ground truth. It will be challenging for those answers that are only part of the tag text.

F1 score (F1) This metric measures the overlap of the predicted answer and the ground truth. We split the answer and ground truth into tokens and compute the F1 score on them.

Path overlap score (POS) When the model predicts an answer from a wrong tag but the text of the answer is identical to the ground truth, the exact match and F1 score will fail. Therefore we introduce path overlap score, a tag level metric that evaluates the accuracy in structure. An HTML document is a DOM tree, so for every tag, there exists a unique path from the root *<HTML>* element to the tag. We compute the path overlap score (POS) between path p_1 and p_2 as following: $\text{POS} = \frac{|P_1 \cap P_2|}{|P_1 \cup P_2|}$, where P_1 and P_2 are the sets of elements in the path p_1 and p_2 respectively. $|\cdot|$ denotes the size of a set.

6.3 Experiment Setup

We train our baselines on the training set and select the best models on the dev set based on the

Models	w/ text	w/ tag	w/ screenshot	DEV			TEST		
				EM	F1	POS	EM	F1	POS
T-PLM (BERT)	√			52.12	61.57	79.74	39.28	49.49	67.68
H-PLM (BERT)	√	√		61.51	**67.04**	82.97	52.61	59.88	76.13
V-PLM (BERT)	√	√	√	**62.07**	66.66	**83.64**	**52.84**	**60.80**	**76.39**
T-PLM (ELECTRA)	√			61.67	69.85	84.15	56.32	72.35	79.18
H-PLM (ELECTRA)	√	√		70.12	74.14	86.33	66.29	72.21	83.17
V-PLM (ELECTRA)	√	√	√	**73.22**	**76.16**	**87.06**	**68.07**	**75.25**	**84.96**

Table 4: Experimental results of various baselines on dev and test sets. EM stands for exact match score, and POS stands for path overlap score.

exact match score. We use uncased BERT-Base and ELECTRA-Large (Clark et al., 2020) as our backbone PLM models. The learning rate is 1e-5. The batch size is 32. We use Adam optimizer with a linear scheduler. For V-PLM, the number of self-attention blocks is 3.

6.4 Results & Discussion

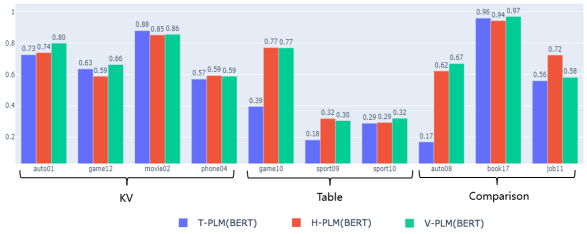


Figure 4: The performance (EM score) comparison of three baselines on 10 different websites on dev set. These websites fall into three categories: *KV*, *Table*, and *Comparison*.

The experimental results are shown in Table 4. We can find that no matter which PLM is used, the more context information (i.e. text, HTML tag, screenshot) yields better performance. Specifically, comparing H-PLM with T-PLM, we find that H-PLM outperforms T-PLM by a large margin. The tag information in H-PLM can implicitly model the visual structure of web pages to some degree. Comparing V-PLM with H-PLM, we find that V-PLM can outperform H-PLM in almost all metrics, which means explicit visual features can provide additional structural information. However, we can also find that the improvement of performance is not very large. This is because the Faster-RCNN toolkit used here is pre-trained on nature images. It may not well apply to screenshots of web pages. In the future, there is a lot of room for exploration of how to make good use of visual information.

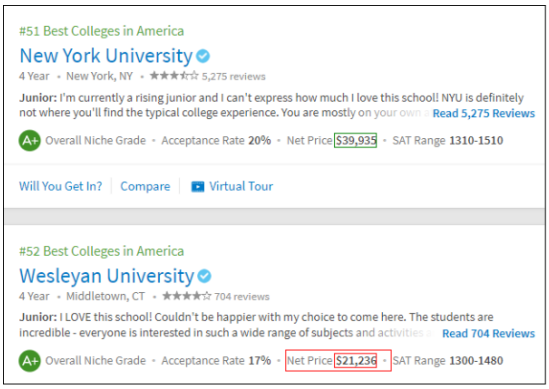
From Table 4, we can also find that ELECTRA-based models consistently outperform BERT-based models. ELECTRA is the best single pre-trained model on text-based MRC tasks, e.g. SQuAD2.0 (Rajpurkar et al., 2018). However, ELECTRA can achieve about 80 EM score on SQuAD2.0, while it can only achieve about 60 EM score on WebSRC. It indicates that WebSRC is still challenging for the current pre-trained language models.

In Figure 4, we further compare the performance of three baseline models on different websites. We find that on three websites, i.e. *game10*, *sport09*, and *auto08*, H-PLM and V-PLM outperform T-PLM with a large margin. Both *game10* and *sport09* fall into the category of *table*, and *auto08* falls into the category of *comparison*. We consider in *comparison* and *table* websites, plain text is not enough to answer questions and more structural information is needed. Generally, to gain good performance on *table* and *comparison* web pages, models should have a good understanding of the global structure of web pages as well as the semantic of contents. From Figure 4, we can also find that among three types of web pages, these models perform worst on *table*.

Method/Metric	#EM	#F1	#POS
T-PLM-SQuAD	29.68	42.91	62.75
T-PLM	54.55	65.28	76.44
H-PLM	61.83	68.24	78.38
V-PLM	**64.71**	**69.26**	**82.81**

Table 5: Results for SQuAD model.

Though the result has shown the difficulty of our dataset, we also wonder about the performance on our dataset of the model that does well on large-scale QA datasets. We fine-tuned a pre-trained BERT-Base model on SQuAD 2.0 dataset, and then use the parameters to initialize our baselines. For H-PLM and V-PLM we still randomly initialize the tag embedding and visual information enhanced self-attention blocks. The SQuAD model we used can achieve an exact match score of 71 in SQuAD. The result is shown in table 5. In the first row,



Question: How much do you have to pay for New York University as tuition?
Answer: $39,935
T-PLM: $21,236 Net Price 1300-1480
H-PLM: $21,236
V-PLM: $21,236

Question: How long is this movie?
Answer: 100 min.
T-PLM: 40: The Temptation of Christ 2020
H-PLM: 40: The Temptation of Christ
V-PLM: 100 min.

Figure 5: Case Study. The true answers are marked by green boxes, and the answers predicted by the model are marked by red boxes.

we report the result of the SQuAD model on our dataset without fine-tuning. The exact match score is only 29.68, which means the texts from HTML are highly different from the normal textual passages. Though the fine-tuned models almost outperform the version without pretraining in all metrics, there is still a large gap between their performance in the textual QA dataset, which means we need more advanced technology to model the HTML structure.

6.5 Case Study

To further analyze the behaviors of our models, we select two images from our dataset and list the predictions made by baselines with a BERT backbone. The result is shown in Figure 5. The image on the left shows information about two universities. The question asked the tuition of the first university but none of the three baselines made the right prediction. T-PLM predicted a longer string other than a raw price because there is no clear boundary of contents in the plain text, while in HTML the tags are natural separators for contents. H-PLM and V-PLM successfully fetched the entire field of *Net Price*, but they failed to model the correspondence between attributes and schools and chose the wrong tuition. The right screenshot is from a movie website, and the question is about the length of the movie. There is no leading text indicating which part would be the length, so the models need to infer the answer from structural information. Both T-PLM and H-PLM predicted the name of the movie, which means they failed to recognize the time information from plain text or HTML. V-PLM can leverage the visual hints and located the right answer. These two examples show that in order to make a comprehensive understanding of web page, a model should be able to understand the visual layout, and group the information correctly according to the spatial structure.

7 Conclusion

In this paper, we introduce WebSRC, a multi-modal dataset for web-based structural reading comprehension with both HTML documents and screenshots. The task is to answer questions about the web pages. We evaluate several baselines on our dataset, and the results showed that incorporating layout features with textual contents is crucial to web understanding, but how to utilize such structural information requires further investigation. We hope this work can push the research on web-based structural reading comprehension forward. In the future, we will go beyond web pages to explore more structural reading comprehension tasks.

Acknowledgments

We sincerely thank the anonymous reviewers for their valuable comments. This work has been supported by the China NSFC Projects (No. 62120106006 and No. 62106142), CCF-Tencent Open Fund and Startup Fund for Youngman Research at SJTU (SFYR at SJTU).

References

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. VQA: Visual Question Answering. In *International Conference on Computer Vision (ICCV)*.

Galal M Binmakhashen and Sabri A Mahmoud. 2019. Document layout analysis: A comprehensive survey. *ACM Computing Surveys (CSUR)*, 52(6):1–36.

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

Peter Buneman. 1997. Semistructured data. In *Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 117–121.

Nilesh Chakraborty, Denis Lukovnikov, Gaurav Maheshwari, Priyansh Trivedi, Jens Lehmann, and Asja Fischer. 2019. Introduction to neural network based approaches for question answering over knowledge graphs. *arXiv preprint arXiv:1907.09361*.

Chia-Hui Chang, Mohammed Kayed, Moheb R Girgis, and Khaled F Shaalan. 2006. A survey of web information extraction systems. *IEEE transactions on knowledge and data engineering*, 18(10):1411–1428.

Danqi Chen. 2018. *Neural reading comprehension and beyond*. Stanford University.

Qiaochu Chen, Aaron Lamoreaux, Xinyu Wang, Greg Durrett, Osbert Bastani, and Isil Dillig. 2021. *Web Question Answering with Neurosymbolic Program Synthesis*, page 328–343. Association for Computing Machinery, New York, NY, USA.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wentau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. Quac: Question answering in context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2174–2184.

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*.

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, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Bhuwan Dhingra, Kathryn Mazaitis, and William W Cohen. 2017. Quasar: Datasets for question answering by search and reading. *arXiv preprint arXiv:1707.03904*.

Matthew Dunn, Levent Sagun, Mike Higgins, V Ugur Guney, Volkan Cirik, and Kyunghyun Cho. 2017. Searchqa: A new q&a dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179*.

Sergio Flesca, Giuseppe Manco, Elio Masciari, Eugenio Rende, and Andrea Tagarelli. 2004. Web wrapper induction: a brief survey. *AI communications*, 17(2):57–61.

Qiang Hao, Rui Cai, Yanwei Pang, and Lei Zhang. 2011. From one tree to a forest: a unified solution for structured web data extraction. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 775–784.

Adam W. Harley, Alex Ufkes, and Konstantinos G. Derpanis. 2015. Evaluation of deep convolutional nets for document image classification and retrieval. In *13th International Conference on Document Analysis and Recognition, ICDAR 2015, Nancy, France, August 23-26, 2015*, pages 991–995. IEEE Computer Society.

Zheng Huang, Kai Chen, Jianhua He, Xiang Bai, Dimosthenis Karatzas, Shijian Lu, and C. V. Jawahar. 2019. ICDAR2019 competition on scanned receipt OCR and information extraction. In *2019 International Conference on Document Analysis and Recognition, ICDAR 2019, Sydney, Australia, September 20-25, 2019*, pages 1516–1520. IEEE.

Mohit Iyyer, Varun Manjunatha, Anupam Guha, Yogarshi Vyas, Jordan Boyd-Graber, Hal Daume, and Larry S Davis. 2017. The amazing mysteries of the gutter: Drawing inferences between panels in comic book narratives. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7186–7195.

Guillaume Jaume, Hazim Kemal Ekenel, and Jean-Philippe Thiran. 2019. FUNSD: A dataset for form understanding in noisy scanned documents. In *2nd International Workshop on Open Services and Tools for Document Analysis, OST@ICDAR 2019, Sydney, Australia, September 22-25, 2019*, pages 1–6. IEEE.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Aniruddha Kembhavi, Minjoon Seo, Dustin Schwenk, Jonghyun Choi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Are you smarter than a sixth grader?

textbook question answering for multimodal machine comprehension. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4999–5007.

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

Nicholas Kushmerick, Daniel S Weld, and Robert Doorenbos. 1997. *Wrapper induction for information extraction*. University of Washington Washington.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794.

David D. Lewis, Gady Agam, Shlomo Argamon, Ophir Frieder, David A. Grossman, and Jefferson Heard. 2006. Building a test collection for complex document information processing. In *SIGIR 2006: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Seattle, Washington, USA, August 6-11, 2006*, pages 665–666. ACM.

Minghao Li, Yiheng Xu, Lei Cui, Shaohan Huang, Furu Wei, Zhoujun Li, and Ming Zhou. 2020. Docbank: A benchmark dataset for document layout analysis. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 949–960.

Peng Li, Wei Li, Zhengyan He, Xuguang Wang, Ying Cao, Jie Zhou, and Wei Xu. 2016. Dataset and neural recurrent sequence labeling model for open-domain factoid question answering. *arXiv preprint arXiv:1607.06275*.

Colin Lockard, Prashant Shiralkar, and Xin Luna Dong. 2019. Openceres: When open information extraction meets the semi-structured web. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3047–3056.

Minesh Mathew, Dimosthenis Karatzas, and C.V. Jawahar. 2021. Docvqa: A dataset for vqa on document images. In *WACV*, pages 2200–2209.

Anand Mishra, Shashank Shekhar, Ajeet Kumar Singh, and Anirban Chakraborty. 2019. Ocr-vqa: Visual question answering by reading text in images. In *ICDAR*.

Ion Muslea, Steve Minton, and Craig Knoblock. 1999. A hierarchical approach to wrapper induction. In *Proceedings of the third annual conference on Autonomous Agents*, pages 190–197.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Siva Reddy, Danqi Chen, and Christopher D Manning. 2019. Coqa: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2016. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137–1149.

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. 2019. Towards vqa models that can read. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8317–8326.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158.

Makarand Tapaswi, Yukun Zhu, Rainer Stiefelhagen, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2016. Movieqa: Understanding stories in movies through question-answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4631–4640.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Semih Yagcioglu, Aykut Erdem, Erkut Erdem, and Nazli Ikizler-Cinbis. 2018. Recipeqa: A challenge dataset for multimodal comprehension of cooking recipes. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1358–1368.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380.

Richard Zanibbi, Dorothea Blostein, and James R Cordy. 2004. A survey of table recognition. *Document Analysis and Recognition*, 7(1):1–16.

Changchang Zeng, Shaobo Li, Qin Li, Jie Hu, and Jianjun Hu. 2020. A survey on machine reading comprehension—tasks, evaluation metrics and benchmark datasets. *Applied Sciences*, 10(21):7640.

Xingyao Zhang, Linjun Shou, Jian Pei, Ming Gong, Lijie Wen, and Daxin Jiang. 2020. A graph representation of semi-structured data for web question answering. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 51–61, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Xu Zhong, Jianbin Tang, and Antonio Jimeno-Yepes. 2019. Publaynet: Largest dataset ever for document layout analysis. In *2019 International Conference on Document Analysis and Recognition, ICDAR 2019, Sydney, Australia, September 20-25, 2019*, pages 1015–1022. IEEE.

Jun Zhu, Zaiqing Nie, Ji-Rong Wen, Bo Zhang, and Wei-Ying Ma. 2005. 2d conditional random fields for web information extraction. In *Proceedings of the 22nd international conference on Machine learning*, pages 1044–1051.

Jun Zhu, Zaiqing Nie, Ji-Rong Wen, Bo Zhang, and Wei-Ying Ma. 2006. Simultaneous record detection and attribute labeling in web data extraction. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 494–503.

A Appendix

A.1 Dataset distribution

The distribution of different website types in various domains is shown in Figure 6. As the figure illustrated, not all domains contain three website types while type KV almost exists in all domains. Websites of type comparison are concentrated in the domain of goods, i.e. *auto*, *book*, in the form of item comparison. Most web pages with type table belong to the domain *sports*, which contain the score data of players.

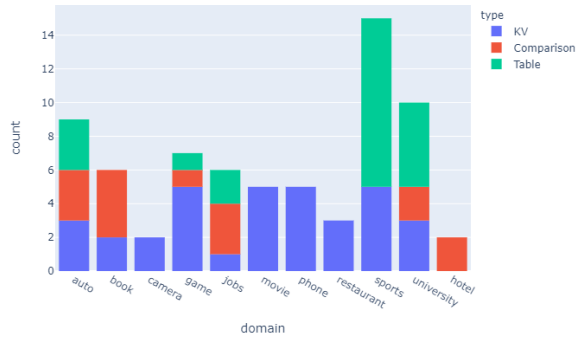


Figure 6: Distribution of different type of websites in different domains.

Figure 7 shows the data distribution in different domains. Domains *auto*, *university* and *sports* account for more than half of the data, and *hotel*, *camera* and *restaurant* are the domains that with least data. This distribution can attribute to the amount of information carried by websites and the amount of information in different domains that are interested by people.

A.2 HTML statistics

We explored the distribution of HTML tags in WebSRC. Figure 8 shows the relative proportion of top 10 frequent HTML tags and top 10 frequent HTML tags containing an answer. Three most common tags are <*div*>, <*td*> and <*span*> on all pages, which are also most frequent tags containing answers. <*div*> and <*span*> are used for separating an area, while <*td*> represents a table cell. This observation indicates that the type of tag may imply the semantics of the content. Though <*div*> is the most frequent tag, <*td*> is much more likely to contain an answer, for the reason that <*div*> is often used in framing the web page while <*td*> is commonly used for presenting a value. The average number of HTML tags in web pages is 177.

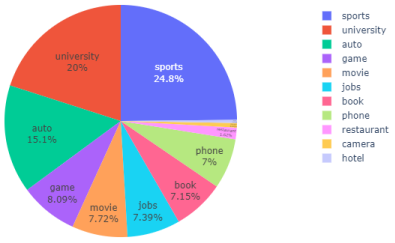


Figure 7: Data distribution in different domains.

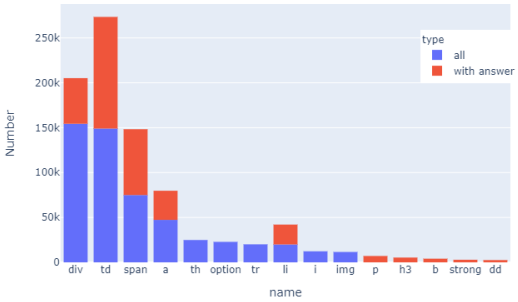


Figure 8: Distribution of HTML tags. The blue is the top 10 HTML tags in all HTML code, and the red is the top 10 HTML tags containing an answer.

The mean depth of HTML DOM trees is 9.8 and the mean depth of tags containing answers is 7.1, which means the upper nodes in the DOM tree would provide more structural information and the lower nodes would contain more specific information.