WebVLN: Vision-and-Language Navigation on Websites

Qi Chen*, Dileepa Pitawela*, Chongyang Zhao*, Gengze Zhou, Hsiang-Ting Chen, Qi Wu†

Australian Institute for Machine Learning, The University of Adelaide
{qi.chen04, dileepa.pitawela, chongyang.zhao, gengze.zhou, tim.chen, qi.wu01}@adelaide.edu.au

Abstract
Vision-and-Language Navigation (VLN) task aims to enable AI agents to accurately understand and follow natural language instructions to navigate through real-world environments, ultimately reaching specific target locations. We recognise a promising opportunity to extend VLN to a comparable navigation task that holds substantial significance in our daily lives, albeit within the virtual realm: navigating websites on the Internet. This paper proposes a new task named Vision-and-Language Navigation on Websites (WebVLN), where we use question-based instructions to train an agent, emulating how users naturally browse websites. Unlike the existing VLN task that only pays attention to vision and instruction (language), the WebVLN agent further considers underlying web-specific content like HTML, which could not be seen on the rendered web pages yet contains rich visual and textual information. Toward this goal, we contribute a dataset, WebVLN-v1, and introduce a novel approach called Website-aware VLN Network (WebVLN-Net), which is built upon the foundation of state-of-the-art VLN techniques. Experimental results show that WebVLN-Net outperforms current VLN and web-related navigation methods. We believe that the introduction of the new WebVLN task and its dataset will establish a new dimension within the VLN domain and contribute to the broader vision-and-language research community. Code is available at: https://github.com/WebVLN/WebVLN.

Introduction
Vision-and-Language Navigation (VLN) (Anderson et al. 2018) aims to seamlessly integrate visual perception and action with language understanding, to enable AI agents to navigate and interact effectively within real-world environments. Interestingly, the resemblance can be found in the virtual online environment, where users might rely on AI agents to assist them in gathering information about certain products even when they can only offer broad and vague instructions such as “how much does a pair of grey and orange striped socks cost”. This extends beyond the boundaries of traditional VLN tasks, including not only vision and instruction (language) but also incorporating the abundant information embedded within webpage like HTML. With this consideration, we introduce an extended VLN task, denoted as Vision-and-Language Navigation on Websites (WebVLN).

Figure 1 shows an example of the WebVLN task. An agent is initiated on the homepage of a website and asked a question $Q$ with an auxiliary description $D$. To respond, the agent is required to intelligently navigate and explore the website, gather information through observation, and finally provide an accurate response/answer $R$ in a free-form sentence.

Figure 1: An example of the WebVLN task. An agent is initiated on the homepage of a website and asked a question $Q$ with an auxiliary description $D$. To respond, the agent is required to intelligently navigate and explore the website, gather information through observation, and finally provide an accurate response/answer $R$ in a free-form sentence.

The Thirty-Eighth AAAI Conference on Artificial Intelligence (AAAI-24)

1165
range of choices for each observed webpage is significantly broader as it contains a vast array of content, features, links, and interactive elements. Each webpage offers multiple avenues for navigation, such as clicking on various links, buttons, and dropdown menus. Second, due to the intrinsic diversity of available choices on each webpage, WebVLN constructs a more intricate and complex navigation graph than traditional VLN, making it nearly impossible to explore all the content on websites by a naive heuristic trial-and-error scheme. Thus, in the WebVLN task, an ideal method should seek to maximise accurate choices while minimising the need for exploration by leveraging varied information available within the webpage.

Due to the lack of an off-the-shelf dataset for WebVLN, we have collected a new WebVLN-v1 dataset to facilitate research in this field. It comprises 8,990 records/paths with 14,825 QA pairs derived from three different shopping websites (aliased as SA, HB and ES). Differing from other VLN datasets (Anderson et al. 2018; Qi et al. 2020b) that only consider the visual inputs of the environment, our WebVLN-v1 incorporates both visual and textual contents extracted from the websites. Furthermore, in comparison to other web-related datasets, such as web navigation (Liu et al. 2018; Xu et al. 2021; Mazumder and Riva 2020; Yao et al. 2022) and web QA (Chang et al. 2022; Hsiao et al. 2022), our WebVLN-v1 seamlessly integrates both navigation and QA environments with question-based instructions, resulting in a unified benchmark.

To tackle the challenging WebVLN task, we propose a new method called Website-aware Vision-and-Language Navigation Network (WebVLN-Net) based on the widely used VLN framework—VLN-CBERT (Hong et al. 2021). Besides the visual input (screenshot) and instruction (question & description), WebVLN-Net considers the underlying HTML of each webpage and extracts elements such as clickable buttons. Upon reaching a “stop” token, our model initiates answering the question using information from both click history and the current “stop” webpage. The evaluation of the model performance is based on the metrics from both VLN and VQA domains. Specifically, for VLN, we consider success rate (SR), oracle success rate (OSR), success rate weighted by path length (SPL), and Trajectory Length (TL), while adopting Wu-Palmer Similarity (WUPS) (Wu and Palmer 1994) for VQA evaluation due to the open-end setting, i.e., generating a free-form sentence as an answer.

In summary, our contributions include:

• A new task - Vision-and-Language Navigation on Websites (WebVLN), where the agent emulates human web browsing behaviours and navigates to a specified target webpage based on the input question and its auxiliary description, subsequently answering the question using information extracted from the target webpage.

• A new WebVLN-v1 dataset, consisting of 8,990 records/paths, and 14,825 question-answer (QA) pairs derived from three different websites, covering both navigation and QA on the web environments.

• A new method, named Website-aware VLN Network (WebVLN-Net), which not only considers the visual input (screenshot) and linguistic instruction but also uses web-specific content (i.e., HTML of the webpage) to enhance decision-making precision.

Related Work

As the WebVLN is a new task, we briefly overview several closely relevant works w.r.t. Vision-and-Language Navigation (VLN) and other web-related navigation and QA tasks.

Vision-and-Language Navigation (VLN) The VLN task (Anderson et al. 2018) extends the vision and language research with sequential action prediction and is one of the most influential tasks in Embodied AI. The research on VLN is dedicated to addressing the alignment of linguistic instructions with visual cues and actions, some work fine-graining the navigation instructions to achieve sub-goal planning (Hong et al. 2020; He et al. 2021a; Zhu et al. 2020), and some concentrate on utilizing object information to identify landmarks from observations (Gao et al. 2021; Qi et al. 2020a, 2021). Temporal information is specifically designed in (Hao et al. 2020; Hong et al. 2021; Chen et al. 2021, 2022; Qiao et al. 2022, 2023; Zhao, Qi, and Wu 2023) to capture long-range dependencies across past observations and actions, which are crucial during navigation. Some methods incorporate external knowledge during navigation. For example, MiniWoB++ (Liu et al. 2018), RUSS (Xu et al. 2021) and FLIN (Mazumder and Riva 2020) cover sites with diverse user instructions from simple tasks to complex ones like booking flights. Many previous works use various methods on these datasets, which, however, depend on Document Object Model (DOM) structure (Jia, Kiros, and Ba 2019; He et al. 2021b) and hence hamper their flexibility. As for web QA, it mimics the human behaviour of posing a question, aggregating information on the webpage, and generating a response. Several benchmarks, such as WebQA (Chang et al. 2022) and ScreenQA (Hsiao et al. 2022), have been proposed. However, they only offer a single webpage for each question. In contrast, we break the boundary between web navigation and web QA by merging them into a unified task called WebVLN, aligning more closely with human behaviour. Moreover, we design a framework (i.e., WebVLN-Net) that can be easily adapted for different websites.
WebVLN Task and Simulator

Problem Definition
As in Figure 1, the WebVLN task requires an agent to follow natural language instructions (i.e., question & description) to navigate from a homepage to a target webpage and answer the question based on the information from both trajectories and the target webpage. Formally, at the beginning of each episode, the agent is given an input question \( Q \) and an auxiliary description \( D \) (e.g., details of the target item) as the “stop” token, which would be predicted if the current state refers to the target webpage. Ideally, given question \( Q \) and auxiliary description \( D \), the agent should predict a response/answer \( R \) based on the contents in the target state \( s_{\text{EOA}} \).

WebVLN Simulator
In this part, we establish a WebVLN simulator on three different shopping websites (aliased as SA, HB, and ES), covering 1,485 products such as socks, fossils and blankets, and mirror the actions of humans. The details are as follows.

Observations To build the simulator, we enable an agent to navigate within a website by interacting with various buttons. During the \( i \)-th webpage \( W^{(i)} \), the simulator generates an RGB image observation \( I^{(i)} \) (i.e., screenshot) that corresponds to the agent’s viewpoint. It is worth noting that, to prevent any instances of mismatch, we provide the entire screenshot of the current webpage, rather than just a partial view within the display window.

Action Space The primary difficulty in simulator implementation lies in defining the action space that depends on the current state. Naturally, we hope that the agent will not jump to the currently unreachable page indiscriminately. Hence, during each step \( t \), the simulator additionally generates a subset of next-step reachable clickable buttons \( B_{t+1} \subseteq B \), where \( B \) contains all clickable buttons present on the website. The agent interacts with the simulator by selecting a new button \( b_{t+1} \in B_{t+1} \). To establish \( B_{t+1} \), the simulator constructs a directed graph for each website, denoted as \( G = \{B, E\} \), where the existence of an edge indicates a navigable transition between two webpages, which is accessible by the agent. In each step, we also incorporate a specific button \( b_{\text{EOA}} \) for the “stop” action.

WebVLN-v1 Dataset Construction

Automatic Path Generation
Following the way for generating path in R2R (Anderson et al. 2018), we sample a target webpage and subsequently construct the shortest path from the homepage to the selected target according to the aforementioned graph \( G \). For each path, we manually check it to determine its reasonability from a human perspective and then discard the unreasonable ones (e.g., clicking the advertisement icon multiple times in succession). Furthermore, we remove paths with fewer than 2 webpage transitions to maintain dataset quality and diversity, resulting in a total sample of 8,990 paths.

LLM-aided Question-Answer Generation
To alleviate the workload on humans, we seek to generate QA pairs with the assistance of Large Language Models (LLMs)\(^2\) based on the multimodal content from the webpage. We depict the details in the following.

Image Data Handling & HTML Cleaning Firstly, we employ BLIP-2 (Li et al. 2023), a large and powerful model for image captioning, to transform the website images into captions, ensuring that the LLMs are able to capture the visual information. After that, another critical step is processing the word list, since the original texts directly from webpages are often disorganised and challenging to work with, which also include irrelevant information, such as messy code. Hence, we systematically adopt a rule-based filtering approach to manually eliminate such information, which enhances the logic and readability of these texts.

Designing Rules for Generating QA Pairs We start to design a series of rules for the LLM to guide its behaviour when generating required QA pairs. The initial rules include:

- Provide 3 questions and their answers that can be directly found from the information provided in the text.
- Ask the first question about price and second about available sizes and the third about material.
- If precise answers cannot be found for those questions, then ask the questions on colours and availability in stock.
- Phrase your questions in a clear and concise manner to ensure they can be accurately answered by the given content.
- Answer should be to the point without additional information.

To mitigate the impact of negative information, we introduce two additional rules as follows.

- The provided text is all from an online shopping website, there is some disturbing information which is irrelevant to the products, such as “sign in”. Make sure your questions and answers will focus on the products themselves.

\(^{1}\)Note that if the question has sufficient information to locate the target webpage, the auxiliary description will remain empty.

\(^{2}\)Here, we use ChatGPT with the gpt-3.5-turbo model.
Table 1: Comparison between our WebVLN-v1 dataset and the related datasets, which cover three different types of environment, i.e., embodied scene, Mobile App, and website. To ensure a more comprehensive comparison, we evaluate the datasets across three dimensions: the environment (Env.), the instruction (Ins.) and the target task. Here, “Temp.” refers to whether the environment would be temporally changed. “Image”, “Text” and “HTML” are the components that environment covers. “Que.” and “Des.” are the abbreviations of question and description, respectively, referring to the type of instruction that the dataset contains. “Ins. Level” means whether the instruction is a high-level statement or a low-level step-by-step command.

<table>
<thead>
<tr>
<th>Env. Type</th>
<th>Dataset</th>
<th>Environment (Env.)</th>
<th>Instruction (Ins.)</th>
<th>Task</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Embodied</td>
<td>R2R</td>
<td>✓ ✓ ✓</td>
<td>✓ ✓ Low</td>
<td>Navigation</td>
<td>21,567</td>
</tr>
<tr>
<td></td>
<td>EQA</td>
<td>✓ ✓ ✓</td>
<td>✓ ✓ High</td>
<td>Navigation + QA</td>
<td>5,281</td>
</tr>
<tr>
<td></td>
<td>REVERIE</td>
<td>✓ ✓</td>
<td>✓ ✓ High</td>
<td>Localise Remote Object</td>
<td>21,702</td>
</tr>
<tr>
<td>Mobile App</td>
<td>PixelHelp</td>
<td>✓ ✓ ✓</td>
<td>✓ ✓ Low</td>
<td>Navigation</td>
<td>187</td>
</tr>
<tr>
<td></td>
<td>MoTIF</td>
<td>✓ ✓ ✓</td>
<td>✓ ✓ High</td>
<td>Navigation</td>
<td>1,125</td>
</tr>
<tr>
<td></td>
<td>META-GUI</td>
<td>✓ ✓ ✓</td>
<td>✓ ✓ High</td>
<td>Dialogue</td>
<td>4,707</td>
</tr>
<tr>
<td>Website</td>
<td>MIND2WEB</td>
<td>✓ ✓ ✓</td>
<td>✓ ✓ Low</td>
<td>Navigation</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>WebQA</td>
<td>✓ ✓</td>
<td>✓ ✓ High</td>
<td>Navigation</td>
<td>741</td>
</tr>
<tr>
<td></td>
<td>ScreenQA</td>
<td>✓ ✓</td>
<td>✓ ✓ High</td>
<td>Question-Answer (QA)</td>
<td>~46,500</td>
</tr>
<tr>
<td></td>
<td>WebVLN-v1</td>
<td>✓ ✓ ✓</td>
<td>✓ ✓ High</td>
<td>Navigation + QA</td>
<td>14,825</td>
</tr>
</tbody>
</table>

Table 1 demonstrates that our WebVLN-v1 dataset is unique, covering rich information (i.e., temporal sequence, image, text and HTML) from the environment and two different types of instructions (i.e., questions and description-statements), while others only focus on them partially. Moreover, our WebVLN-v1 is able to support both navigation and question-answering tasks, rather than other website-related datasets that can only support one of them.

WebVLN-v1 Dataset Analysis

**WebVLN-v1 Dataset vs. Related Datasets**

We compare our WebVLN-v1 dataset with the most relevant datasets w.r.t. Embodied AI, Mobile App, and Website. Specifically, for Embodied AI datasets, we consider R2R (Anderson et al. 2018), REVERIE (Qi et al. 2020b) and EQA (Das et al. 2018), where the first two are the widely used vision-and-language navigation (VLN) datasets while the last one is a famous embodied question answering dataset. Regarding App-based datasets, we compare PixelHelp (Li et al. 2020), MoTIF (Burns et al. 2022) and META-GUI (Sun et al. 2022). As for the website, we consider seven datasets for a comprehensive comparison, including MiniWoB++ (Liu et al. 2018), RUSS (Xu et al. 2021), FLIN (Mazumder and Riva 2020), WebShop (Yao et al. 2022), MIND2WEB (Deng et al. 2023), WebQA (Chang et al. 2022), and ScreenQA (Hsiao et al. 2022).

Table 1 demonstrates that our WebVLN-v1 dataset is unique, covering rich information (i.e., temporal sequence, image, text and HTML) from the environment and two different types of instructions (i.e., questions and description-statements), while others only focus on them partially. Moreover, our WebVLN-v1 is able to support both navigation and question-answering tasks, rather than other website-related datasets that can only support one of them.

**WebVLN-v1 Statistics**

**Word Cloud** In Figure 2, we visualise the questions, auxiliary descriptions, and answers of our WebVLN-v1 dataset as Venn-style word cloud (Coppersmith and Kelly 2014). The size of each word is the harmonic mean of its count.

**Lengths of Question, Description, Answer and Path** Figure 3 exhibits the distributions of question length, de-
For the WebVLN task, we propose a new model, called WebVLN-Net, which is based on a widely used vision-and-language navigation framework VLN$\odot$BERT (Hong et al. 2021). As shown in Figure 4, our model contains three main components: initialisation, navigation and answering. Specifically, we first initialise the state and context tokens by a pre-trained BERT model. Subsequently, we input these initialised language tokens, along with the screenshot and button tokens extracted from the current webpage, into the navigation component. This process iterates until the target webpage is reached. Last, the answering head in the answering component generates the final answer.

**State and Context Initialisation**  In initialisation ($t = 0$), our model receives a word sequence comprising the classification token [CLS], the separation token [SEP], and the language tokens $V$ extracted from both the question $Q$ and the auxiliary description $D$. Here, [CLS] and [SEP] are predefined tokens in BERT. Similar to VLN$\odot$BERT (Hong et al. 2021), the [CLS] token is used to aggregate relevant vision-language cues from the input sequence. In this context, we define the embedded [CLS] token as the initial state representation $s_0$. We update it during the whole training phase, ensuring it could be aware of the entire navigation and question-answering tasks. The process can be defined as

$$s_0 = \text{Init}([\text{CLS}], Q, [\text{SEP}], D),$$

where the $\text{Init}(\cdot)$ indicates the initialisation process.

**Web Navigation**  We adapt the model proposed in (Hong et al. 2021) to incorporate the learning of navigation and the concurrent selection of clickable buttons. As shown in Figure 4, at each time step, the network takes four different token sets as input: the preceding state token $s_{t-1}$, the language tokens $V_t$, the screenshot tokens $I_t$, and the button tokens $B_t$. Specifically, we “patchify” the screenshot as a set of image patches and convert them to the tokens $I_t$ by using a Transformer-based image encoder. Likewise, as each button consists of an image and a description (i.e., “alt” in HTML), we introduce a button encoder that contains an image encoder and a text encoder to derive their corresponding tokens. After that, we concatenate tokens associated with the same button, followed by projecting the concatenated token back to its original dimension using a linear projection layer. Subsequently, we put all the tokens into a multi-layer Transformer to obtain an action probability $p_t$:

$$s_t, p_t = \text{Nav}(s_{t-1}, V_t, I_t, B_t).$$

Here, the $\text{Nav}(\cdot)$ refers to the navigation process in each step. Notably, the set of button tokens $B_t$ involves an End Of Action ([EOA]) token, selected when the agent reaches the target webpage. The state would be updated to $s_{t+1}$.

In navigation steps ($t > 0$), the state token $s_t$, screenshot tokens $I_t$, and button tokens $B_t$ employ self-attention across the entire input sequence, while the language tokens $V$ only serve as the keys and values in Transformer. We regard the language tokens derived from the model’s initialisation step as a good representation of both the question and auxiliary description ($Q$ & $D$), obviating the necessity for subsequent encoding in later stages and saving computational resources.

**Question Answering**  To generate the final answer, we introduce an Answering Head, which is a $M$-layer Transformer decoder. Considering an open-ended question-answering setting, our objective is to generate the answer as a free-form sentence autoregressively. Mathematically,

$$R = \text{Ans}(s_{\text{EOA}}),$$

where $\text{Ans}(\cdot)$ refers to the answering process. Here, $R$ is the predicted answer consisting of $L$ words (i.e., $R = \{w_l\}_{l=1}^L$) and $s_{\text{EOA}}$ denotes the last state aforementioned above.
Training For navigation, we train our network using an imitation learning (IL) objective. Concretely, our agent navigates on the ground-truth trajectory by following the teacher actions and computes a cross-entropy loss for each decision made. Formally, we minimise the navigation loss function, which can be formulated for each specific sample by

\[ L_{\text{nav}} = -\sum_t a_t \log(p_t) - \eta \sum_t a^*_t \log(p_t), \]  

where \( a_t \) is the sampled action and \( a^*_t \) is the teacher action. Here, \( \eta \) represents a coefficient used to weigh the IL loss.

The training of the Answering Head employs the autoregressive Teacher Forcing (Williams and Zipser 1989) approach, wherein the prediction at each step should maximise the likelihood of the subsequent token:

\[ L_{\text{ans}} = \sum_{l=1}^{L} -\log p(w_l|w_{<l}, s_{[\text{EOA]}}), \]  

where \( w_l \) indicates the \( l \)-th token in the answer, and \( w_{<l} \) denotes the tokens before \( w_l \). \( L \) is the total number of tokens. The final loss function can be defined as

\[ L = L_{\text{nav}} + \lambda L_{\text{ans}}, \]  

where \( \lambda \) is the weighting hyper-parameter.

Experiments
Evaluation Metrics and Baselines
Metrics for Navigation Following the evaluation metrics in R2R (Anderson et al. 2018), we assess the shortest path distance in the navigation graph \( G \) between the finally located webpage of the agent and the target webpage. We consider an episode to be a success if the agent stops on the target webpage, \( i.e., \) success rate (SR), and its variant – oracle success rate (OSR). We also measure the navigation performance by considering the path length, \( i.e., \) the success rate weighted by Path Length (SPL) and Trajectory Length (TL).

Metrics for QA For question-answering, we follow the open-ended setting that seeks to generate a free-form natural language sentence to answer the given question, which is more flexible and practical than regarding QA as a classification problem. Thus, rather than an exact accuracy between predicted and ground-truth answers, we adopt Wu-Palmer Similarity (WUPS) (Wu and Palmer 1994), aiming to quantify the semantic differences between a predicted answer and the ground truth. WUPS assigns a value ranging between 0 and 1, reflecting their degree of similarity. Following (Maliowski and Fritz 2014), we set thresholds of the WUPS as 0.9 and 0.0 separately with a scaling factor of 0.1, whereby scores below the threshold are proportionally adjusted.

Baselines We evaluate the performance of both navigation and QA by comparing the results with baselines. For a comprehensive comparison, we consider two different types of baselines, \( i.e., \) traditional VLN and web-related navigation. Specifically, we employ VLN \( \bowtie \) BERT (Hong et al. 2021) (randomly initialised and initialised by LXMERT (Tan and Bansal 2019)) as the VLN baseline, which is widely used in the VLN task. Notably, to well evaluate the performance, we seek to adapt VLN \( \bowtie \) BERT to our task and dataset with minimal changes. Specifically, following its original design, we take as inputs the linguistic instruction (question & description) and screenshot. The model would predict the intermediate navigation action and finally answer the question by incorporating the same QA head as ours. As for the web-related navigation, we consider a state-of-the-art (SoTA) instruction-finetuned foundation model WebGUM (Furuta et al. 2023), which is built upon the T5 model (Raffel et al.
The Thirty-Eighth AAAI Conference on Artificial Intelligence (AAAI-24)

Table 2: We compare our method with VLN ∩ BERT and WebGUM, which are the widely used VLN model and SoTA multimodal web navigation foundation model, respectively. VLN ∩ BERT and VLN ∩ BERT* are randomly initialised and initialised by LXMERT, respectively. WebGUM and WebGUM† denote the models based on T5-small and T5-base separately.

Table 3: Ablation study on the test set of WebVLN-v1 dataset. Q, D and I are the input question, auxiliary description and screenshot, respectively. The notations d and e are the button description/text and the button image separately.

Table 4: Zero-shot by using LLMs. * denotes using GPT4 while others are ChatGPT (gpt-3.5-turbo).

Zero-shot using LLMs To further test the difficulty of our WebVLN task and the WebVLN-v1 dataset, we conduct a zero-shot evaluation setting for the popular large language models (LLMs). We randomly select 201 samples from the validation set and test on two LLM-as-agent pipelines, AgentBench (Liu et al. 2023) and NavGPT (Zhou, Hong, and Wu 2023). For AgentBench, we adopt the same prompt from the WS (Web Shopping) task in the paper and delete the one-shot example, test on the gpt-3.5-turbo model. For NavGPT, we modify the VLN task description into a web shopping task description and test on both gpt-3.5-turbo and gpt-4 models. All the formatted observations are replaced with the WebVLN-v1 sample observation. From Table 4, the zero-shot performance of these methods falls short of reaching human-level performance. It is necessary for the ongoing advancement of intelligent agents, as current SoTA LLMs are far from perfect performance in our task. Moreover, our WebVLN can also serve as a metric to gauge such progress.

Conclusion In this paper, we propose a novel task, Vision-and-Language Navigation on Websites (WebVLN), extended from conventional VLN. It seeks to enable an agent to answer the user’s questions via navigating/exploring the websites and integrating useful information. To support research in this new task, we collect a new WebVLN-v1 dataset and design a baseline method called Website-aware Vision-and-Language Navigation Network (WebVLN-Net). The experiments demonstrate the effectiveness of our WebVLN-Net. Moreover, we perform a zero-shot evaluation of LLM-based methods using the WebVLN-v1 dataset, where the performance is far from saturation, highlighting the utility of our WebVLN-v1 as a benchmark to assess progress in this field.
References


