NavHint: Vision and Language Navigation Agent with a Hint Generator

Yue Zhang
Michigan State University
zhan1624@msu.edu

Quan Guo
Sichuan University
guoquan@scu.edu.cn

Parisa Kordjamshidi
Michigan State University
kordjams@msu.edu

Abstract

Existing work on vision and language navigation mainly relies on navigation-related losses to establish the connection between vision and language modalities, neglecting aspects of helping the navigation agent build a deep understanding of the visual environment. In our work, we provide indirect supervision to the navigation agent through a hint generator that provides detailed visual descriptions. The hint generator assists the navigation agent in developing a global understanding of the visual environment. It directs the agent's attention toward related navigation details, including the relevant sub-instruction, potential challenges in recognition and ambiguities in grounding, and the targeted viewpoint description. To train the hint generator, we construct a synthetic dataset based on landmarks in the instructions and visible and distinctive objects in the visual environment. We evaluate our method on the R2R and R4R datasets and achieve state-of-the-art on several metrics. The experimental results demonstrate that generating hints not only enhances the navigation performance but also helps improve the interpretability of the agent's actions.

1 Introduction

In many real-world applications, it is a crucial skill for an intelligent agent to perceive the visual environment and interact with humans using natural language. The Vision and Language Navigation (VLN) task (Anderson et al., 2018) is one of the popular problems in this direction that has attracted significant attention from computer vision, natural language processing, and robotic communities (Li et al., 2022; Fried et al., 2018; Francis et al., 2022).

With the increasing popularity of the VLN task, many neural navigation models (Hong et al., 2020c; Chen et al., 2021; Hao et al., 2020) have been proposed. One line of research is to strengthen the connection of the vision and language modalities (Ma et al., 2019; Hong et al., 2020a; Li et al., 2021). However, the majority of these efforts learn the connection mainly supervised by navigation performance, such as the distance to the destination, the orientation selection (heading and elevation), and the similarity between the given instruction and the trajectory. While this helps teach the agent to navigate, it does not directly enforce learning comprehensive textual and visual semantics. In fact, learning visual semantics in the environment is crucial not only for successfully completing navigation tasks but also for the effective communication with humans. For instance, the navigation agent should correctly locate the navigation progress based on the current visual views. Moreover, the navigation agent needs to adopt a global perspective of the environment to investigate whether the navigable viewpoints include the relevant landmarks or whether the instruction is ambiguous. In any case, the agent should be able to describe its targeted viewpoint. Expecting the navigation agent to obtain the above understanding solely through navigation-related signals is challenging, and the intermediate guidance is necessary.

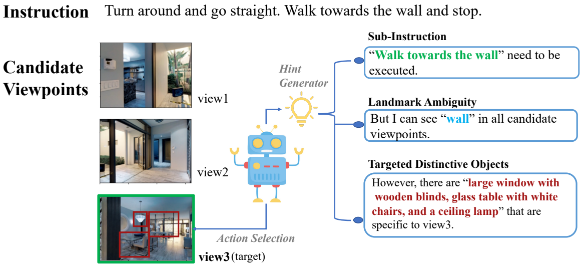

Instruction
Turn around and go straight. Walk towards the wall and stop.
Candidate Viewpoints
view1
view2
view3 (target)
Hint Generator
Action Selection
Sub-Instruction
"Walk towards the wall" need to be executed.
Landmark Ambiguity
But I can see "wall" in all candidate viewpoints.
Targeted Distinctive Objects
However, there are "large window with wooden blinds, glass table with white chairs, and a ceiling lamp" that are specific to view3.

Figure 1: Given the instruction and three candidate viewpoints, the navigation agent with the assistance of the hint generator, produces descriptions of the visual environment with three key elements: sub-instruction, landmark ambiguity and targeted distinctive objects.

To this end, we introduce a hint generator for the VLN agent (NavHint), aiming to generate visual descriptions that serve as indirect supervision

Findings of the Association for Computational Linguistics: EACL 2024, pages 92–103
March 17-22, 2024 ©2024 Association for Computational Linguistics

to help the navigation agent obtain a better understanding of the visual environment (as depicted in Fig. 1). When the agent navigates at each step, the hint generator concurrently produces visual descriptions that are consistent with the agent's action decision. The hints are designed based on the rationale underlying the navigation process, including three aspects: *Sub-instruction*, *Landmark Ambiguity* and *Targeted Distinctive Objects*. Specifically, at each navigation step, **first**, the hint generator encourages the agent to report its navigation progress by specifying which part of the sub-instruction it is executing based on the current visual environment. As depicted in Fig. 1, the sub-instruction "walk towards the wall" needs to be executed. **Second**, the hint generator directs the agent to have a global view of the entire environment and recognize the landmarks mentioned in the instruction from all candidate viewpoints. The agent is tasked with identifying potential challenges by assessing the visibility of the landmarks and comparing the landmarks shared among viewpoints. For instance, in the given example, the landmark "wall" is ambiguous as it appears in multiple views. **Third**, in scenarios where challenges exist, the hint generator guides the agent in describing the distinctive visual objects that only appear in the targeted viewpoint, such as "*large window with wooden blinds*" in view3 in Fig 1. This aids the agent in deeply looking into the details of its selected viewpoint while globally comparing it to other candidates.

The hint generator is designed as a Transformer-based decoder that leverages visual output from the navigation agent to produce corresponding hints. This hint generator can be plugged into any VLN agent as a language model conditioned on the VLN models. To train the hint generator, we propose a synthetic navigation hint dataset based on Room2Room (R2R) (Anderson et al., 2018) dataset. Our dataset provides hints for each step of the trajectory in the R2R dataset. Each hint description includes sub-instruction, landmark ambiguity, and targeted distinctive objects introduced above. The dataset serves as an extra supervision to train the navigation agent and the hint generator jointly. Besides, our constructed dataset can be utilized to explicitly analyze the navigation agent's grounding ability by assessing the quality of generated hints.

In summary, our contributions are as follows:

1. We leverage a language model conditioned on the VLN models to design a hint generator that can be plugged into any VLN agent. This hint generator helps the agent develop a comprehensive understanding of the visual environment.
2. We construct a synthetic hint dataset to provide the agent with visual descriptions at each navigation step. The dataset serves as an indirect supervision for jointly training the navigation agent and the hint generator.
3. We show that the hint generation improves the agent's navigation performance on the R2R and R4R datasets. We also provide a detailed analysis of the agent's grounding ability by examining the quality of the generated hints, thereby improving the interpretability of the agent's decisions.

2 Related Work

Navigation Instruction Following Anderson et al. (2018) first extended the instruction following to the photo-realistic simulated environments. Subsequent studies have emerged with an emphasis on enhancing navigation performance through multimodal learning (Hong et al., 2020a; Wang et al., 2023b; Zhang and Kordjamshidi, 2022a; An et al., 2021; Zhang et al., 2021), map representation learning (Hong et al., 2023; Chen et al., 2022a; An et al., 2023), or graph-based explorations (Zhu et al., 2021; Wang et al., 2021; Chen et al., 2022b). One line of effort has been to provide auxiliary reasoning tasks or pre-training proxy tasks to guide the navigation agent to learn textual and visual representations (Zhu et al., 2020; Chen et al., 2021; Hao et al., 2020; Qiao et al., 2022; Zhang and Kordjamshidi, 2022b). AuxRN (Zhu et al., 2020) proposes four auxiliary reasoning tasks to gain knowledge of the navigation map and the consequences of actions. However, most of those methods acquire the textual and visual semantics from a wayfinding perspective during navigation, which may be insufficient for agents to understand the visual environment comprehensively. We address this issue with our proposed hint generator that offers visual descriptions to guide the navigation agent in learning visual semantics.

Language-Capable VLN Agent A few studies attempt to design language-capable VLN agents to improve the agent's grounding ability. Most of the work encourages the navigation agent to reproduce the original instruction. For example, LANA (Wang et al., 2023a) devises an agent that executes human-written navigation commands and provides route descriptions. Similarly, one of the tasks in AuxRN (Zhu et al., 2020) is to retell the

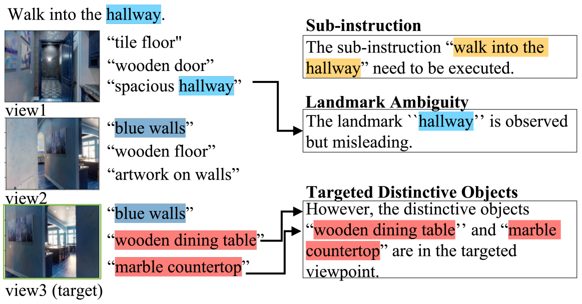

Walk into the hallway.
"tile floor"
"wooden door"
"spacious hallway"
view1
"blue walls"
"wooden floor"
"artwork on walls"
view2
"blue walls"
"wooden dining table"
"marble countertop"
view3 (target)
Sub-instruction
The sub-instruction "walk into the hallway" need to be executed.
Landmark Ambiguity
The landmark ``hallway'' is observed but misleading.
Targeted Distinctive Objects
However, the distinctive objects "wooden dining table'' and "marble countertop'' are in the targeted viewpoint.

Figure 2: Navigation Hint Dataset. An example of a navigation hints with the landmark ambiguity of "*Missing Landmarks*". The sub-instruction is"walk into the hallway"(🟧), and the landmark "hallway" (🟦) in the instruction is observed in the view1 rather than target view3, which can potentially mislead the navigation agent. The target distinctive objects "wooden dining table" and "marble countertop."(🟥) are then provided. "Blue walls" (🟦) is non-distinctive as it appears in both view2 and view3.

trajectory. However, these approaches have limitations because the original instruction can sometimes be inaccurate and confusing, as suggested in the VLN-Trans (Zhang and Kordjamshidi, 2023). Forcing the agent to reproduce the same instruction in such cases can undermine the agent's grounding ability. Instead of only focusing on the original instruction, our proposed hint generator produces visual descriptions from a global perspective, thereby enhancing the agent's understanding of the visual environment and improving its grounding ability.

3 Method

In the VLN problem setting, the agent is given a natural language instruction, denoted as $W = \{w_1, w_2, \cdots, w_l\}$, l is the length of the sentence. At each navigation step, the agent perceives panoramic views with 36 [1] discrete images. There are n candidate viewpoints that can be navigated to, denoted as $I = \{I_1, I_2, \cdots, I_n\}$. This task aims to generate a trajectory following the given instruction. In the following section, we first present our constructed navigation hint dataset. Then, we introduce the hint generator. The navigation hint dataset is used to train the navigation agent and the hint generator jointly.

3.1 Navigation Hint Dataset

The purpose of constructing the navigation hint dataset is to provide supervision for the hint generator to generate detailed visual description. The navigation hint dataset is automatically generated based on instruction and trajectory pairs from the R2R dataset (Anderson et al., 2018). For every step of the trajectory, we provide hints that mainly include three key elements, as described below.

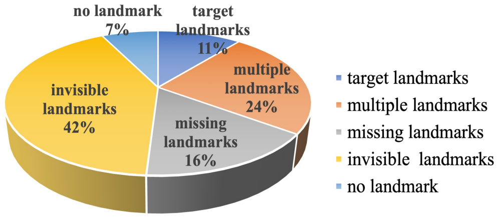


Figure 3: Statistics of different categories of landmark ambiguity.

Sub-instruction is the first part of the hint that pinpoints to the relevant part of the instruction (sub-instruction) to be processed at the current step. We obtain the sub-instructions and their corresponding viewpoints from the FGR2R (Hong et al., 2020b) dataset, which provides human annotations of sub-instructions and the aligned viewpoints.

After obtaining the sub-instruction at each step, we insert it into our hint template, which is "*The {sub-instruction} needs to be executed.*". Guiding the navigation agent to detect the related sub-instruction at each step is crucial since it effectively assists the agent in tracking its navigation progress.

Landmark Ambiguity is the second part of the hint that describes the commonalities across multiple views that can result in ambiguity during navigation. This part of hint is achieved by examining the shared landmarks mentioned in the instruction among the candidate viewpoints.

To automatically generate this part of the hint for building the dataset, we first use spaCy[2] to extract noun phrases from sub-instruction and use them as landmarks. Then, we extract visual objects in each candidate viewpoint using MiniGPT-4 (Zhu et al., 2023)[3] with a two-step textual prompting. We choose visual objects generated by MiniGPT-4 instead of Matterport3D object annotations because Matterport3D objects are pretty limited, with only 40 object categories like "*doors*", "*walls*", and "*floors*". These generic objects are not sufficient for resolving landmark ambiguity. Moreover, the absence of attribute annotations in Matterport3D poses a challenge for landmark disambiguation, such as the differences between "*wooden table*" and "*glass table*". In contrast, MiniGPT-4 can generate such detailed attribute descriptions.

[1] 12 headings and 3 elevations with 30-degree intervals.

[2] https://spacy.io/

[3] https://minigpt-4.github.io/

Ambiguity Category	Description	Hints
Target Landmarks	Landmarks only appear in the target.	The {landmarks} are observed.
Multiple Landmarks	Landmarks are visible in multiple viewpoints including the target viewpoint.	The {landmarks} are observed in multiple viewpoints.
Missing Landmarks	Landmarks are visible in other viewpoints except for the target viewpoint.	The{landmarks} are misleading.
Invisible Landmark	Landmarks are not visible in all viewpoints	The{landmarks} are not observed.
No Landmarks	No landmarks in sub-instruction. (*e.g.* "*make a right turn*", "*turn left*", and "*go straight*")	∅

Table 1: Landmark Ambiguity. The col#1 and col#2 show the categories of landmark ambiguity and the corresponding descriptions. The col#3 shows the template for generating the hint for each category.

Specifically, for each candidate viewpoint, we feed MiniGPT-4 with the viewpoint image, asking "*Describe the details of the image.*" and then "*List the objects in the image*". The generated text is in free form, and we post-process it to retrieve a list of extracted object descriptions. After obtaining textual landmark names and visual objects, we examine the shared landmarks among the candidate viewpoints. The presence of shared landmarks can pose ambiguity for the navigation agent. We categorize the ambiguity into: *Target Landmarks*, *Multiple Landmarks*, *Missing Landmarks*, *Invisible Landmarks* and *No Landmark.* and their descriptions are in Table 1. Fig. 3 shows the statistics of ambiguity of our navigation hint dataset. Most cases are "*Invisible Landmarks*" or "*Multiple Landmarks*", which is consistent with the argument in VLN-trans (Zhang and Kordjamshidi, 2023) that invisible and non-distinctive landmarks cause issues for the navigation agent in following instructions.

After identifying the category of landmark ambiguity, we construct this part of the hint using the corresponding templates in col #3 of Table 1. Identifying landmark ambiguity requires the navigation agent to ground the mentioned landmark names in the instruction to the visual objects in all candidate viewpoints. Guiding the navigation agent to identify such detailed ambiguities can help enhance its understanding of the connection between the instruction and the entire visual environment.

Targeted Distinctive Objects is the third part of the hint that describes the distinctive visual objects specific to the targeted view. The agent should be able to justify its decision by describing the distinction of the targeted view. We follow the approach of obtaining distinctive objects in the VLN-Trans (Zhang and Kordjamshidi, 2023) that compares the visual objects in the targeted and other candidate viewpoints. The distinctive objects are the ones that exclusively appear in the targeted viewpoint and do not appear in other views.

The hint template for targeted distinctive objects is "*However, {the comma-separated list of distinctive object names} are in the targeted view.*". We use 3 distinctive objects at most. If the cases belong to the challenge of "*Target Landmark*", there is no need to provide extra distinctive objects since the landmark is already exclusive to the targeted viewpoint. Describing distinctive objects is important to obtain a global understanding of the visual environment by highlighting the differences between the targeted viewpoint and other candidate viewpoints.

We collect hint for each step of trajectory to construct our navigation hint dataset. More details are in Appendix A.1.

3.2 VLN Agent with a Hint Generator

We propose a hint generator that can be plugged into any navigation agent easily. We use VLN↻BERT (Hong et al., 2020c) as the base model to illustrate our method but noted that the hint generator is compatible with most of the current agents. Fig. 4 shows the model architecture.

Text Encoder We use BERT (Vaswani et al., 2017) to obtain initial text representation of instruction, denoted as $X = [x_1, x_2, \cdots, x_l]$.

Vision Encoder We follow previous works to concatenate image and relative orientation features as vision features for each candidate viewpoint. Specifically, we extract the image features from ResNet-152 (He et al., 2016) pre-trained on the Places365 dataset (Zhou et al., 2017). The orientation features are derived from the relative heading denoted as α and the elevation denoted as β. The orientation features are represented as $[\sin\alpha; \cos\alpha; \sin\beta; \cos\beta]$. The vision features are then passed through an MLP (Multilayer Perception) of Vision Encoder to obtain vision representation for each candidate viewpoint, denoted as $[v_1, v_2, \cdots, v_n]$.

Navigation Agent VLN↻BERT is a cross-modal Transformer model. Besides text and vision representations, a state representation is introduced in the model to store history information recurrently, which is denoted as S. At the t-th navigation step, the text representation X, the visual representation V_t and state representation S_t are input into cross-modal Transformer layers, as follows,

$$\hat{X}, \hat{S}_t, \hat{V}_t = Cross\_Attn(X, [S_t; V_t]), \quad (1)$$

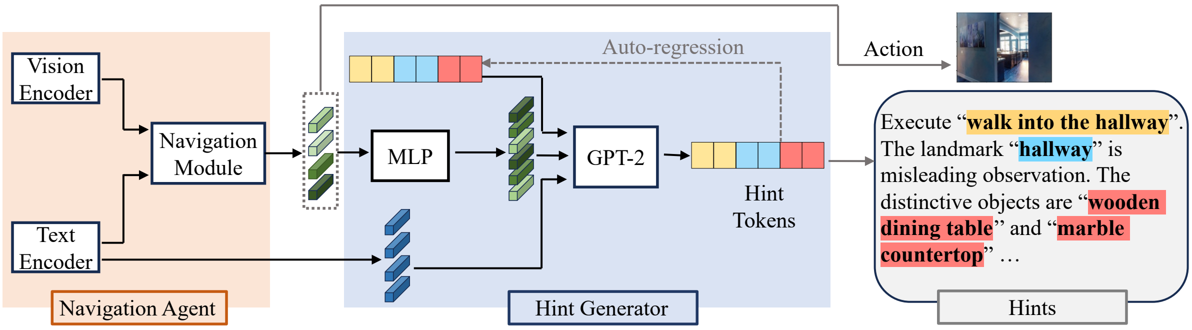

Vision Encoder
Text Encoder
Navigation Module
Navigation Agent
MLP
GPT-2
Auto-regression
Hint Tokens
Hint Generator
Action
Execute "walk into the hallway". The landmark "hallway" is misleading observation. The distinctive objects are "wooden dining table" and "marble countertop" …
Hints

Figure 4: Model Architecture. We introduce a hint generator designed to help the navigation agent acquire a deep understanding of the visual environment. The weighted vision representations (🟩), used as image prefix, and the instruction text representation, used as instruction prefix (🟦), are input into a GPT2 decoder. The decoder generates hints during navigation at each step. The hints include the three parts of sub-instruction (🟧), landmark ambiguity (🟦), and target distinctive objects (🟥).

where $\hat{X}$, $\hat{S}_t$, and $\hat{V}_t$ are the learnt contextual text, state representation, and visual representations, respectively. Then we apply attention layer between state representation $\hat{S}_t$ contextual vision representations $\hat{V}_t$ as follows,

$$S_{t+1}, a_t = Attn(k = \hat{V}_t, q = \hat{S}_t, v = \hat{V}_t), \quad (2)$$

where S_{t+1} is the updated state representation that is passed to the next steps to convey the history. a_t is the attention score over the navigable views and serves as the action probability of the current step.

Hint Generator Inspired by the idea of prefix engineering (Mokady et al., 2021) that uses the image representation as the prefix of the text for the image captioning task, we employ a decoder language model (LM) and use the contextual visual representation of the navigation agent and the original instruction as the prefix. However, unlike the previous work, rather than just using one image as the prefix, we input all images of candidate viewpoints to encourage the hint generator to learn the global relations among views.

Formally, we denote the hint at the i-th navigation step as $C^i = \{c_1^i, c_2^i, \cdots, c_j^i\}$, where j is the length of the hint. Different from LANA (Wang et al., 2023a) that generates route description after navigation, our hint generator provides a more in-depth visual description at each step. Our approach requires the agent to possess a global and deep visual understanding, which can be learnt through the supervision from our navigation hint dataset explained in Section 3.1. We obtain the LM representation of the original instruction W and the hint C as $X' = \{x_1', x_2', \cdots, x_l'\}$ and $c = \{c_1, c_2, \cdots, c_j\}$ respectively. Since the semantic structure of our auto-generated dataset can be easily captured, we use a 1.5B-parameters decoder LM (GPT-2 large) in the hint generator. Note that any larger decoder language model in the GPT series can be employed.

We use the instruction text representation X' as the instruction prefix representation. We use the weighted vision representations output from the navigation agent as the image prefix representation. The weighted vision representation is obtained using action probability and the contextual vision representations as $\hat{\hat{V}}_t = a_t * \hat{V}_t$. Then we simply employ an MLP to map $\hat{\hat{V}}_t$ to LM token space. We denote such MLP as F. We obtain prefix embedding that is mapped from visual representation $\hat{V}$ as follows,

$$p_1, \cdots, p_k = F(\hat{\hat{V}}_t), \quad (3)$$

where k is the prefix length, and p is the image prefix representation. We concatenate the representation of image prefix p and instruction prefix X', and combine them with the text representation of hint C. The hint generator only decodes the hint in an auto-regressive manner at each step. During training, the parameters of both of MLP and the LM in the hint generator and the navigator are updated. The training objective is to maximize the likelihood of the next hint token. The following equation shows the loss of generating the j-th token of the hint at the i-th step.

$$L_{hint} = -\sum_{i,j} \log p_\theta(c_j^i | p_1^i, \cdots, p_k^i, \\ x_1', \cdots, x_l', c_j^i, \cdots, c_{j-1}^i). \quad (4)$$

Training and Inference for the VLN Agent For the navigation, we train the navigation with a mixture of Imitation Learning (IL) and Reinforcement

		Validation Unseen					Test Unseen		
	Method	NE ↓	SR ↑	SPL↑	sDTW↑	nDTW↑	NE ↓	SR ↑	SPL ↑
1	Seq-to-Seq (Anderson et al., 2018)	7.81	0.22	–	–	–	7.85	0.20	0.18
2	Self-Monitor (Ma et al., 2019)	5.52	0.45	0.32	–	–	5.67	0.48	0.35
3	AuxRN (Ma et al., 2019)	5.63	0.51	0.46	–	–	–	–	–
4	VLN↻BERT (Hong et al., 2020c)	3.93	0.63	0.57	–	–	4.09	0.63	0.57
5	HAMT (ViT) (Chen et al., 2021)	3.97	0.66	0.61	–	–	3.93	0.65	**0.60**
6	LANA (Wang et al., 2023a)	–	0.66	0.60	–	–	–	0.64	0.59
7	VLN-SIG (ViT) (Li and Bansal, 2023)	3.37	0.68	0.62	0.59	0.70	–	0.65	**0.60**
8	VLN-trans (Zhang and Kordjamshidi, 2023)	3.34	**0.69**	0.63	0.60	0.70	3.94	**0.66**	**0.60**
9	EDrop* (Tan et al., 2019)	5.49	0.55	0.47	0.42	0.58	5.60	0.51	0.49
10	EDrop + Hint. (NavHint)	5.44	0.55	0.47	0.44	0.60	5.47	0.53	0.49
11	VLN↻BERT^{++} (Zhang and Kordjamshidi, 2023)	3.40	0.67	0.61	0.58	0.69	4.02	0.63	0.58
12	VLN↻BERT^{++} + Hint. (NavHint)	**3.23**	**0.69**	**0.65**	**0.61**	**0.72**	4.00	0.65	**0.60**

Table 2: Experimental results on R2R dataset. The best results are in bold font. VLN↻BERT^{++} is the improved version of VLN↻BERT by pre-training the cross representations using a larger dataset (see Sec 4.2). ViT: uses Vision Transformer representations. Hint.: uses our hint generator.

Learning (RL) (Tan et al., 2019). It consists of the cross-entropy loss of the predicted probability distribution against the ground-truth action and a sampled action from the predicted distribution to learn the designed rewards. In summary, the navigation loss is as follows,

$$L_{nav} = -\sum_t -\alpha_t^* log(p_t^\alpha) - \lambda \sum_t \alpha_t^s log(p_t^\alpha), \quad (5)$$

where λ is the hyperparameter to balance the two components, α_t^* is the teacher action for IL, and α_t^s is sample action for RL. We jointly train the navigation agent with hint generator using the following objective,

$$L = L_{hint} + L_{nav}. \quad (6)$$

During inference of navigation, we use greedy search to select an action with the highest probability at each navigation step to generate a trajectory. To generate hint, we utilize the trained weighted visual representation and the original instruction text representation as prompts and employ a greedy search approach to generate the hints.

4 Experiment

4.1 Dataset and Evaluation Metrics

Dataset We evaluate our approach on R2R (Anderson et al., 2018) and R4R datasets (Jain et al., 2019), which are built upon Matterport3D simulator (Anderson et al., 2018). R2R includes $21,567$ instructions and $7,198$ trajectories. R4R is an extension of R2R to combine the two adjacent tail-to-head trajectories in R2R. The visual environments in unseen sets are excluded in the training sets.

Evaluation Metrics Three main metrics are used to evaluate navigation wayfinding performance (Anderson et al., 2018). (1) Navigation Error (NE) (2) Success Rate (SR) (3) Success Rate Weighted Path Length (SPL). Another three metrics measure the fidelity between the predicted and the ground-truth trajectories. (4) Coverage Weighted by Length Score (CLS) (Jain et al., 2019) (5) normalized Dynamic Time Warping (nDTW) (Ilharco et al., 2019) (6) Normalized Dynamic Time Warping weighted by Success Rate (sDTW). More details are in Appendix A.2 and A.3.

4.2 Implementation Details

We use pre-trained VLN↻BERT^{++} (Zhang and Kordjamshidi, 2023) to initialize our navigation model. VLN↻BERT^{++} further trains the pre-trained weights in VLN↻BERT (Hong et al., 2020c; Hao et al., 2020) on a large image-text-action dataset including RXR (Ku et al., 2020), Marky-mT5 (Wang et al., 2022), and SyFis (Zhang and Kordjamshidi, 2023). The dimensions of both BERT and GPT text representations are 768-d. In the training, we conducted 300K iterations on an NVIDIA RTX GPU (20 hours), with a batch size of 8 and a learning rate of $1e-5$. λ in Eq. 5 is 0.2. We set the maximum prefix length for each image as 10 for the hint generator and the number of generated tokens as 80. The best model is selected according to performance on val unseen split. Please check our code [4] for the implementation.

4.3 Experimental Results

Table 2 shows the performance on validation unseen and test of the R2R dataset in a *single-run setting* where the navigation agent traverses without *backtracking* and *pre-exploring*. To verify the adaptability of our approach, we evaluate it using both LSTM-based and Transformer-based navigation agents. Since Transformer-based methods

[4] https://github.com/HLR/NavHint.git

	Method	**NE↓**	**SR↑**	**SPL↑**	**CLS↑**	**sDTW↑**
1	OAAM (Qi et al., 2020)	13.80	0.29	0.18	0.34	0.11
2	RelGraph (Hong et al., 2020a)	7.55	0.35	0.25	0.37	0.18
3	NvEM (An et al., 2021)	6.80	0.38	0.28	0.41	0.20
4	VLN↻BERT (Hong et al., 2020c)	6.48	0.43	0.32	0.42	0.21
5	CITL (Liang et al., 2022)	6.42	0.44	0.35	0.39	0.23
6	VLN-Trans (Zhang and Kordjamshidi, 2023)	**5.87**	**0.46**	**0.36**	**0.45**	**0.25**
7	VLN↻BERT^{++} (Zhang and Kordjamshidi, 2023)	6.33	0.44	0.34	0.43	0.23
8	VLN↻BERT^{++} + Hint. (NavHint)	6.04	**0.46**	**0.36**	**0.45**	**0.25**

Table 3: Results on R4R validation unseen dataset.

are pre-trained on large vision-language datasets and have a more complex model architecture, they achieve a higher performance than LSTM-based methods. For the LSTM-based model, we use EDrop (Tan et al., 2019) which uses CLIP (Radford et al., 2021) visual representations without augmented data during training. For the Transformer-based model, we use the VLN↻BERT^{++} (row#11) as the baseline.

Row#1 to row#3 in Table 2 show other LSTM-based methods and row#4 to row#8 are the SOTA Transformer-based methods. Row#9 shows the performance of the LSTM baseline EDrop. Row#10 shows the results after equipping the EDrop with our designed hint generator. The improved sDTW and nDTW on the validation unseen proves that the hint generator helps the navigation agent follow the instructions. Moreover, our hint generator on top of the VLN↻BERT^{++} (row#12) significantly improves both wayfinding metrics (SP and SPL) and fidelity metrics (sDTW and nDTW) of the baseline model, indicating that our hint generator not only assists the agent in reaching the correct destination but also encourages the agent to follow the original instructions. Improving both LSTM-based and Transformer-based navigation agents shows the generalization ability of the navigation agent with our designed hint generator.

Table 3 shows the results on the unseen validation of the R4R dataset. We use VLN↻BERT^{++} as our baseline model (row#7). Row#1 to row#3 are using LSTM-model, and row#4 to row#6 are using Transformer-based models. The result of our method (row#8) shows that we can improve SPL, sDTW, and CLS, that is, improving both the wayfinding and fidelity of the baseline models. These results are consistent with the improvements on the R2R dataset. Though the VLN-Trans (row#6) (SOTA) is very competitive, we additionally provide hints that can be used for explicitly analyzing the agent's decisions instead of implicit sub-instruction learning designed in VLN-Trans.

Model	**Val Seen**		**Val Unseen**	
	`Bleu-1`	`Bleu-4`	`Bleu-1`	`Bleu-4`
EDrop + Hint. (ours)	0.74	0.62	0.72	0.60
VLN↻BERT^{++}+ Hint. (ours)	**0.76**	**0.64**	**0.74**	**0.62**

Table 4: `Bleu` score for the generated sub-instruction on the R2R dataset.

Method	**Hints**				**Val Unseen**		
	Sub.	**L-A.**	**TD-Obj.**	**Obj.**	**SR↑**	**SPL↑**	**nDTW↑**
Baseline					0.665	0.607	0.685
1	✔				0.671	0.612	0.690
2		✔			0.673	0.613	0.687
3			✔		0.677	0.624	0.702
4				✔	0.676	0.621	0.698
5	✔	✔			0.674	0.614	0.709
6	✔	✔		✔	0.681	0.632	0.694
7	✔	✔	✔		**0.692**	**0.647**	**0.724**

Table 5: Ablation study, where Baseline is VLN↻BERT^{++}. Sub.:sub-instruction; L-A.:Landmark Ambiguity; TD-Obj: Target Distinctive Objects. Obj:Top-3 objects.

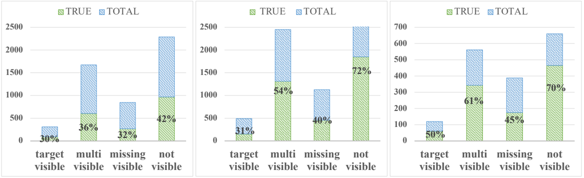
4.4 Ablation Study

Table 5 reports the ablation analysis. From row#1 to row#3, we individually include sub-instruction, landmark ambiguity, and targeted distinctive objects to the hint. All navigation performance metrics improve gradually compared to the baseline. In another experiment (row#4), we attempt to describe the visual environment by identifying only top-3 recognized objects (using MiniGPT-4) in the targeted viewpoint without differing them from other viewpoints. The navigation results still improve, indicating that visual descriptions of the objects benefit the overall navigation performance. Row#5 shows that combining sub-instruction and landmark ambiguity further improves the baseline, particularly in the nDTW metric. In row#6, when we combine sub-instruction, landmark ambiguity and top-3 objects, we observe improvement in the goal-related metrics (SR and SPL), but the model's ability to faithfully follow the instruction is somewhat compromised (lower nDTW). The best result is obtained when we replace the above top-3 objects with distinctive ones (row#7), indicating our designed hint's effectiveness in describing the targeted view from a global perspective.

4.5 Generated Hints Analysis

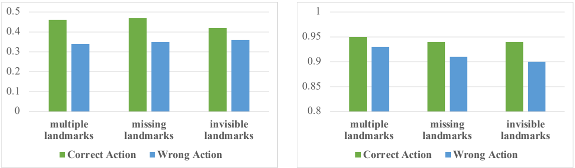
In this section, we assess the content of each part of the generated hints on the R2R validation dataset to analyze agent's grounding ability.

Sub-instruction Analysis We use `Bleu` score (Papineni et al., 2002) as an evaluation metric to assess whether the navigation agent can identify sub-instruction accurately. We conduct experiments on both LSTM-based and Transformer-based navigation agents, as shown in Table 4. The generated sub-instruction from the Transformer-based navigation agent can obtain a relatively high `Bleu` score compared to the LSTM-based agent. This result



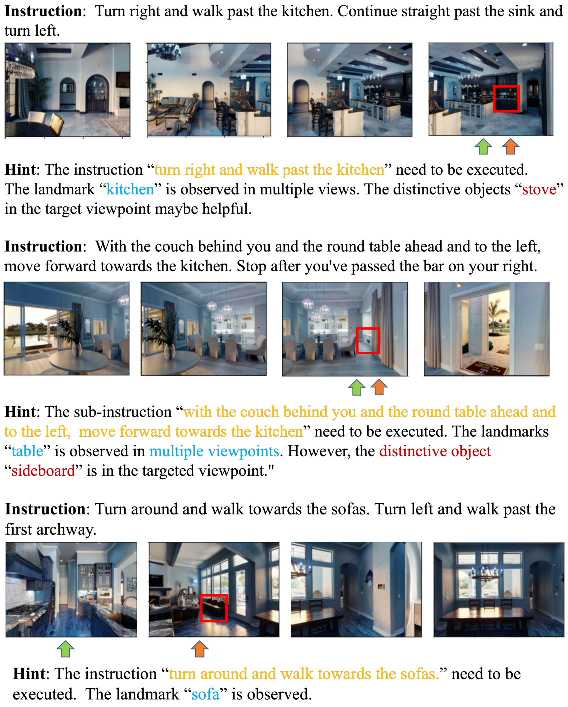
(a) EDrop+Hint. (b) VLN↻BERT^{++}+Hint. (c) Correct Sub.

Figure 5: Accuracy of the generated landmark ambiguity. Sub.: Sub-instruction.



(a) Exact Matching (b) Object Matching

Figure 6: Accuracy of the generated distinctive objects for each landmark ambiguity in the targeted viewpoint.



Instruction: Turn right and walk past the kitchen. Continue straight past the sink and turn left.

Hint: The instruction "turn right and walk past the kitchen" need to be executed. The landmark "kitchen" is observed in multiple views. The distinctive objects "stove" in the target viewpoint maybe helpful.

Instruction: With the couch behind you and the round table ahead and to the left, move forward towards the kitchen. Stop after you've passed the bar on your right.

Hint: The sub-instruction "with the couch behind you and the round table ahead and to the left, move forward towards the kitchen" need to be executed. The landmarks "table" is observed in multiple viewpoints. However, the distinctive object "sideboard" is in the targeted viewpoint."

Instruction: Turn around and walk towards the sofas. Turn left and walk past the first archway.

Hint: The instruction "turn around and walk towards the sofas." need to be executed. The landmark "sofa" is observed.

Figure 7: Qualitative examples. The green and orange arrows show the ground-truth and the predicted viewpoints, respectively.

demonstrates that a more robust navigation agent achieves a stronger alignment between the instruction and visual modality for identifying the relevant part of the instruction to track the progress.

Landmark Ambiguity Analysis We assess the accuracy of four categories of landmark ambiguity in the generated hints. Specifically, We extract the part of the landmark ambiguity from the generated hint and check its accuracy in the visual environment. In Figure 4, the TOTAL in the y-axis shows the total number of navigation steps that include each ambiguity category, shown on the x-axis. The TRUE (green) indicates the percentage of navigation steps when the corresponding ambiguity truly exists. We evaluate both LSTM-based and Transformer-based agents, and the result shows that Transformer-based agents can achieve higher accuracy of landmark ambiguity. We conclude that accurate landmark ambiguity detection is positively correlated with better navigation performance. In Figure 4(c), we evaluate the generated hint for the examples in which the sub-instruction is generated correctly, as indicated by a `Bleu-4` score of 1.0. In those examples, the accuracy of identifying each category of landmark ambiguity is also higher. This result shows accurately locating the sub-instruction positively impacts landmark ambiguity detection.

Targeted Distinctive Objects Analysis We report the accuracy of identifying the targeted distinctive objects in the generated hints when landmark ambiguity exists, as shown in Fig. 6. The generated hints are from the model of VLN↻BERT^{++} with our designed hint generator. We provide two types of comparisons, exact phrase matching and object token matching while performing both wrong and right actions. Exact matching evaluates the detection of distinctive object tokens and the attribute descriptions in the whole referring phrase. Object matching only evaluates the detection of distinctive object tokens. The result shows that the accuracy in generating distinctive objects is generally higher when the action is correct than when it is wrong. Also, the agent tends to generate distinctive objects that align with its targeted viewpoint, as indicated by an accuracy exceeding 90%, even when the action is incorrect. The lower accuracy of exact matching also aligns with the fact that generating the whole referring expression, including the correct attributes, is more challenging.

4.6 Qualitative Examples

Fig. 7 demonstrates a few examples of the generated descriptions. The first two examples show successful cases where the agent makes a correct decision. The first example shows the agent can accurately identify the sub-instruction and notice the ambiguous landmark "kitchen". Then, it correctly pinpoints the distinctive object "stove", which only appears in the target viewpoint. In fact, our *targeted distinctive object* design can help connect the specific object (e.g. stove, refrigerator, counter table) to more general scene objects (e.g. kitchen). Also, the second example shows the agent accurately points out the "table" in the instruction that appears in multiple viewpoints and refers to the "sideboard" in the target viewpoint. The third example shows a failure case in which the agent makes a wrong decision. The sub-instruction is correctly identified, but the agent should turn around towards the counter table and proceed to the sofa rather than walk to the sofa directly. This further indicates that our descriptor pushes the model to focus on landmarks directly and ignore the directions and motions in the instruction. Despite this, our model can generate a description consistent with its selection. More examples are in the Appendix A.4.

5 Conclusion

In this paper, we equip the navigation agent with a hint generator to generate visual descriptions during navigation, which helps the agent's understanding of the visual environment. To train the hint generator, we create a navigation hint dataset that provides comprehensive supervision for training the agent. During navigation, the agent generates natural language descriptions about its visual environment at each step, including comparing various views and explaining ambiguities in recognizing the target destination. Empirical results show that detailed visual description generation improves both navigation performance and the interpretability of actions taken by the navigation agent.

6 Limitations

We mainly summarize the following limitations. First, although we employ the GPT2 language decoder, more recent and powerful GPT-series language decoders are now available and could be utilized. Exploring these advanced language decoders could potentially enhance the performance of our approach. Second, we do not include more advanced vision representations, such as ViT representation, to train the navigation agent. We can surpass other methods using ResNet, but it would be interesting to experiment with those different visual representations to generate better hints. Third, utilizing object visual descriptions from MiniGPT-4 may entail hallucination issues, which is a general challenge of VLMs. However, in our specific usage of MiniGPT4, we barely face this issue in the experiments.

7 Acknowledgement

This project is supported by the National Science Foundation (NSF) CAREER award 2028626 and partially supported by the Office of Naval Research (ONR) grant N00014-20-1-2005 and grant N00014-23-1-2417. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation nor the Office of Naval Research. We thank all reviewers for their thoughtful comments and suggestions.

References

Dong An, Yuankai Qi, Yan Huang, Qi Wu, Liang Wang, and Tieniu Tan. 2021. Neighbor-view enhanced model for vision and language navigation. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 5101–5109.

Dong An, Hanqing Wang, Wenguan Wang, Zun Wang, Yan Huang, Keji He, and Liang Wang. 2023. Etpnav: Evolving topological planning for vision-language navigation in continuous environments. *arXiv preprint arXiv:2304.03047*.

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton Van Den Hengel. 2018. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3674–3683.

Peihao Chen, Dongyu Ji, Kunyang Lin, Runhao Zeng, Thomas Li, Mingkui Tan, and Chuang Gan. 2022a. Weakly-supervised multi-granularity map learning for vision-and-language navigation. *Advances in Neural Information Processing Systems*, 35:38149–38161.

Shizhe Chen, Pierre-Louis Guhur, Cordelia Schmid, and Ivan Laptev. 2021. History aware multimodal transformer for vision-and-language navigation. *Advances in neural information processing systems*, 34:5834–5847.

Shizhe Chen, Pierre-Louis Guhur, Makarand Tapaswi, Cordelia Schmid, and Ivan Laptev. 2022b. Think global, act local: Dual-scale graph transformer for vision-and-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16537–16547.

Jonathan Francis, Nariaki Kitamura, Felix Labelle, Xiaopeng Lu, Ingrid Navarro, and Jean Oh. 2022. Core challenges in embodied vision-language planning. *Journal of Artificial Intelligence Research*, 74:459–515.

Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. 2018. Speaker-follower models for vision-and-language navigation. *Advances in Neural Information Processing Systems*, 31.

Weituo Hao, Chunyuan Li, Xiujun Li, Lawrence Carin, and Jianfeng Gao. 2020. Towards learning a generic agent for vision-and-language navigation via pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13137–13146.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Yicong Hong, Cristian Rodriguez, Yuankai Qi, Qi Wu, and Stephen Gould. 2020a. Language and visual entity relationship graph for agent navigation. *Advances in Neural Information Processing Systems*, 33:7685–7696.

Yicong Hong, Cristian Rodriguez-Opazo, Qi Wu, and Stephen Gould. 2020b. Sub-instruction aware vision-and-language navigation. *arXiv preprint arXiv:2004.02707*.

Yicong Hong, Qi Wu, Yuankai Qi, Cristian Rodriguez-Opazo, and Stephen Gould. 2020c. A recurrent vision-and-language bert for navigation. *arXiv preprint arXiv:2011.13922*.

Yicong Hong, Yang Zhou, Ruiyi Zhang, Franck Dernoncourt, Trung Bui, Stephen Gould, and Hao Tan. 2023. Learning navigational visual representations with semantic map supervision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3055–3067.

Gabriel Ilharco, Vihan Jain, Alexander Ku, Eugene Ie, and Jason Baldridge. 2019. General evaluation for instruction conditioned navigation using dynamic time warping. *arXiv preprint arXiv:1907.05446*.

Vihan Jain, Gabriel Magalhaes, Alexander Ku, Ashish Vaswani, Eugene Ie, and Jason Baldridge. 2019. Stay on the path: Instruction fidelity in vision-and-language navigation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1862–1872.

Alexander Ku, Peter Anderson, Roma Patel, Eugene Ie, and Jason Baldridge. 2020. Room-across-room: Multilingual vision-and-language navigation with dense spatiotemporal grounding. *arXiv preprint arXiv:2010.07954*.

Jialu Li and Mohit Bansal. 2023. Improving vision-and-language navigation by generating future-view image semantics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10803–10812.

Jialu Li, Hao Tan, and Mohit Bansal. 2021. Improving cross-modal alignment in vision language navigation via syntactic information. *arXiv preprint arXiv:2104.09580*.

Jialu Li, Hao Tan, and Mohit Bansal. 2022. Envedit: Environment editing for vision-and-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15407–15417.

Xiwen Liang, Fengda Zhu, Yi Zhu, Bingqian Lin, Bing Wang, and Xiaodan Liang. 2022. Contrastive instruction-trajectory learning for vision-language navigation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 1592–1600.

Chih-Yao Ma, Jiasen Lu, Zuxuan Wu, Ghassan AlRegib, Zsolt Kira, Richard Socher, and Caiming Xiong. 2019. Self-monitoring navigation agent via auxiliary progress estimation. *arXiv preprint arXiv:1901.03035*.

Ron Mokady, Amir Hertz, and Amit H Bermano. 2021. Clipcap: Clip prefix for image captioning. *arXiv preprint arXiv:2111.09734*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Yuankai Qi, Zizheng Pan, Shengping Zhang, Anton van den Hengel, and Qi Wu. 2020. Object-and-action aware model for visual language navigation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part X 16*, pages 303–317. Springer.

Yanyuan Qiao, Yuankai Qi, Yicong Hong, Zheng Yu, Peng Wang, and Qi Wu. 2022. Hop: history-and-order aware pre-training for vision-and-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15418–15427.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

Hao Tan, Licheng Yu, and Mohit Bansal. 2019. Learning to navigate unseen environments: Back translation with environmental dropout. *arXiv preprint arXiv:1904.04195*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Hanqing Wang, Wenguan Wang, Wei Liang, Caiming Xiong, and Jianbing Shen. 2021. Structured scene memory for vision-language navigation. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 8455–8464.

Su Wang, Ceslee Montgomery, Jordi Orbay, Vighnesh Birodkar, Aleksandra Faust, Izzeddin Gur, Natasha Jaques, Austin Waters, Jason Baldridge, and Peter Anderson. 2022. Less is more: Generating grounded navigation instructions from landmarks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15428–15438.

Xiaohan Wang, Wenguan Wang, Jiayi Shao, and Yi Yang. 2023a. Lana: A language-capable navigator for instruction following and generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19048–19058.

Zun Wang, Jialu Li, Yicong Hong, Yi Wang, Qi Wu, Mohit Bansal, Stephen Gould, Hao Tan, and Yu Qiao. 2023b. Scaling data generation in vision-and-language navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12009–12020.

Yue Zhang, Quan Guo, and Parisa Kordjamshidi. 2021. Towards navigation by reasoning over spatial configurations. *arXiv preprint arXiv:2105.06839*.

Yue Zhang and Parisa Kordjamshidi. 2022a. Explicit object relation alignment for vision and language navigation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 322–331.

Yue Zhang and Parisa Kordjamshidi. 2022b. Lovis: Learning orientation and visual signals for vision and language navigation. *arXiv preprint arXiv:2209.12723*.

Yue Zhang and Parisa Kordjamshidi. 2023. Vln-trans: Translator for the vision and language navigation agent. *arXiv preprint arXiv:2302.09230*.

Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. 2017. Places: A 10 million image database for scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 40(6):1452–1464.

Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. 2023. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*.

Fengda Zhu, Xiwen Liang, Yi Zhu, Qizhi Yu, Xiaojun Chang, and Xiaodan Liang. 2021. Soon: Scenario oriented object navigation with graph-based exploration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12689–12699.

Fengda Zhu, Yi Zhu, Xiaojun Chang, and Xiaodan Liang. 2020. Vision-language navigation with self-supervised auxiliary reasoning tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10012–10022.

A Appendix

A.1 Statistics of the VLN Hint Dataset

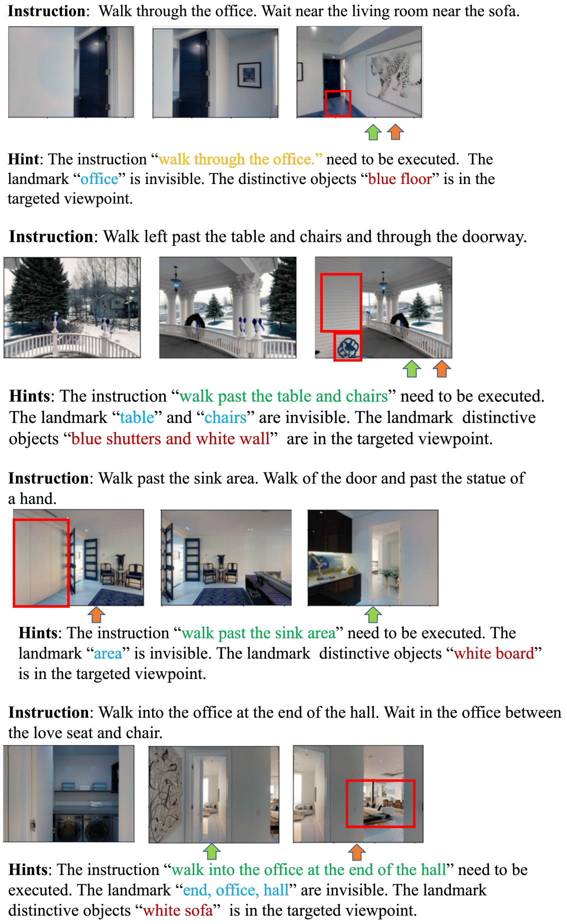
We built VLN explanation dataset upon R2R dataset. We split our explanation dataset into train, validation seen, and validation unseen sets according to R2R. We create explanation for each navigation step of trajectory given the corresponding instruction. For train set, there are $4,675$ trajectories, and we create $69,969$ explanation in 61 visual scenes. For validation seen set, there are 340 trajectories, and we create $5,175$ explanations in 61 visual scenes. For validation unseen set, there are 783 trajectories, and we create $11,664$ explanations in 11 visual scenes.

A.2 Dataset

We evaluate our approach on R2R (Anderson et al., 2018) and R4R datasets (Jain et al., 2019), which are built upon Matterport3D simulator (Anderson et al., 2018). R2R includes $21,567$ instructions and 7198 trajectories. The dataset has been partitioned into four sets: train (61 scenes, $14,039$ instructions), validation seen (61 scenes, $1,021$ instructions), validation unseen (11 scenes, $2,349$ instructions), and test unseen sets (18 scenes, $4,173$ instructions). R4R is an extension of R2R to combine the two adjacent tail-to-head trajectories in R2R. It contains three sets: train (61 scenes, $233,613$ instructions), validation seen (61 scenes, $1,035$ instructions), validation unseen (11 scenes, $45,162$ instructions). The scenes in unseen sets are not trained.

A.3 Evaluation Metrics

Three main metrics are used to evaluate navigation wayfinding performance (Anderson et al., 2018): (1) Navigation Error (NE): the mean of the shortest path distance between the agent's final position and the goal destination. (2) Success Rate (SR): the percentage of the predicted final position being within 3 meters from the goal destination. (3) Success Rate Weighted Path Length (SPL): normalizes success rate by trajectory length. Another three metrics are used to measure the fidelity between the predicted and the ground-truth trajectory. (4) Coverage Weighted by Length Score (CLS) (Jain et al., 2019) (6) nDTW (Ilharco et al., 2019): Normalized Dynamic Time Warping: penalizes deviations from the ground-truth trajectories. (6) Normalized Dynamic Time Warping weighted by Success Rate (sDTW) (Ilharco et al., 2019): penalizes deviations from the ground-truth trajectories and also considers the success rate.



Instruction: Walk through the office. Wait near the living room near the sofa.

Hint: The instruction "walk through the office." need to be executed. The landmark "office" is invisible. The distinctive objects "blue floor" is in the targeted viewpoint.

Instruction: Walk left past the table and chairs and through the doorway.

Hints: The instruction "walk past the table and chairs" need to be executed. The landmark "table" and "chairs" are invisible. The landmark distinctive objects "blue shutters and white wall" are in the targeted viewpoint.

Instruction: Walk past the sink area. Walk of the door and past the statue of a hand.

Hints: The instruction "walk past the sink area" need to be executed. The landmark "area" is invisible. The landmark distinctive objects "white board" is in the targeted viewpoint.

Instruction: Walk into the office at the end of the hall. Wait in the office between the love seat and chair.

Hints: The instruction "walk into the office at the end of the hall" need to be executed. The landmark "end, office, hall" are invisible. The landmark distinctive objects "white sofa" is in the targeted viewpoint.

Figure 8: More qualitative examples. The green and orange arrows show the ground-truth and the predicted viewpoints, respectively.

A.4 More Qualitative Examples

We present additional qualitative examples in this section. The first three are successful cases where the navigation agent makes correct actions, and the hint generator accurately generates sub-instruction, landmark ambiguity and distinctive objects in the instruction. The last two examples are failure cases. Despite incorrect actions, the agent still generates accurate distinctive objects within its selected viewpoint. The failures might come from inaccuracies in landmark extraction, which subsequently affect ambiguity checking.