VLN-Trans: Translator for the Vision and Language Navigation Agent

Yue Zhang
Michigan State University
zhan1624@msu.edu

Parisa Kordjamshidi
Michigan State University
kordjams@msu.edu

Abstract

Language understanding is essential for the navigation agent to follow instructions. We observe two kinds of issues in the instructions that can make the navigation task challenging: 1. The mentioned landmarks are not recognizable by the navigation agent due to the different vision abilities of the instructor and the modeled agent. 2. The mentioned landmarks are applicable to multiple targets, thus not distinctive for selecting the target among the candidate viewpoints. To deal with these issues, we design a translator module for the navigation agent to convert the original instructions into easy-to-follow sub-instruction representations at each step. The translator needs to focus on the recognizable and distinctive landmarks based on the agent's visual abilities and the observed visual environment. To achieve this goal, we create a new synthetic sub-instruction dataset and design specific tasks to train the translator and the navigation agent. We evaluate our approach on Room2Room (R2R), Room4room (R4R), and Room2Room Last (R2R-Last) datasets and achieve state-of-the-art results on multiple benchmarks.

1 Introduction

Vision-and-Language Navigation (VLN) (Anderson et al., 2018) task requires an agent to understand and follow complex instructions to arrive at a destination in a photo-realistic simulated environment. This cross-domain task attracts researchers from the communities of computer vision, natural language processing, and robotics (Gu et al., 2022; Wu et al., 2021; Francis et al., 2022).

To solve the VLN task, one streamline of methods is to build the connections between text and vision modalities by grounding the semantic information dynamically (Hong et al., 2020a; Qi et al., 2020a; An et al., 2021; Zhang and Kordjamshidi, 2022a). However, we observe two types of instructions that make the grounding in the VLN task quite challenging.
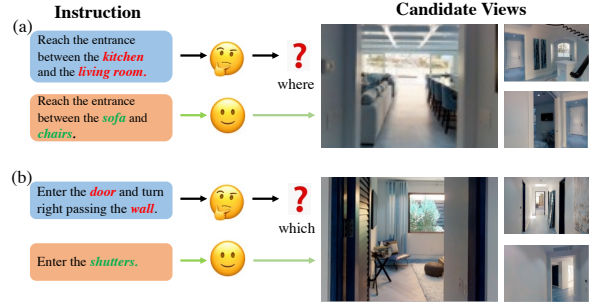

Figure 1: Two types of instructions that make the grounding in the VLN task challenging: (a) unrecognizable landmarks, (b) nondistinctive landmarks. In candidate views, the largest image shows the target view.

challenging. **First**, the instruction contains landmarks that are not recognizable by the navigation agent. For example, Figure 1(a), the agent can only see the "sofa", "table" and "chair" in the target viewpoint, based on the learned vision representations (He et al., 2016; Ren et al., 2015; Dosovitskiy et al., 2020). However, the instructor mentions landmarks of the "living room" and "kitchen" in the instruction, based on their prior knowledge about the environment, such as relating "sofa" to "living room". Given the small size of the dataset designed for learning navigation, it is hard to expect the agent to gain the same prior knowledge as the instructor. **Second**, the instructions contain the landmarks that can be applied to multiple targets, which causes ambiguity for the navigating agent. In Figure 1(b), the instruction "enter the door" does not help distinguish the target viewpoint from other candidate viewpoints since there are multiple doors and walls in the visual environment. As a result, we hypothesize those types of instructions cause the explicit and fine-grained grounding to be less effective for the VLN task, as appears in (Hong et al., 2020b; Zhang et al., 2021) that use sub-instructions and in (Hong et al., 2020a; Hu et al., 2019; Qi et al., 2020a; Zhang and Kordjamshidi, 2022a) that use object-level representations.

To address the aforementioned issues, the main

idea in our work is to introduce a translator module in the VLN agent, named VLN-trans, which takes the given instruction and visual environment as inputs and then converts them to easy-to-follow sub-instructions focusing on two aspects: 1) *recognizable* landmarks based on the navigation agent's visualization ability. 2) *distinctive* landmarks that help the navigation agent distinguish the targeted viewpoint from the candidate viewpoints. Consequently, by focusing on those two aspects, the translator can enhance the connections between the given instructions and the agent's observed visual environment and improve the agent's navigation performance.

To train the translator module, we propose a Synthetic Fine-grained Sub-instruction dataset called SyFiS. The SyFiS dataset consists of pairs of the sub-instructions and their corresponding viewpoints, and each sub-instruction contains a motion indicator and a landmark. We select a motion verb for an action based on our action definitions according to the relative directions between source and target viewpoints; To obtain the landmarks, we first use Contrastive Language-Image Pretraining (CLIP) (Radford et al., 2021), a vision & language pre-trained model with powerful cross-modal alignment ability, to detect the objects in each candidate viewpoint as the recognizable landmarks. Then we select the distinctive one among recognizable landmarks that only appears in the target viewpoint. We train the translator in a contrastive manner by designing positive and negative sub-instructions based on whether a sub-instruction contains distinctive landmarks.

We design two tasks to pre-train the translator: *Sub-instruction Generation* (SG) and *Distinctive Sub-instruction Learning* (DSL). The SG task enables the translator to generate the correct sub-instruction. The DSL task encourages the translator to learn effective sub-instruction representations that are close to positive sub-instructions with distinctive landmarks and are far from the negative sub-instructions with irrelevant and nondistinctive landmarks. Then we equip the navigation agent with the pre-trained translator. At each navigation step, the translator adaptively generates easy-to-follow sub-instruction representations for the navigation agent based on given instructions and the agent's current visual observations. During the navigation process, we further design an auxiliary task, *Sub-instruction Split* (SS), to optimize the trans-

lator module to focus on the important portion of the given instruction and generate more effective sub-instruction representations.

In summary, our contributions are as follows:

1. We propose a translator module that helps the navigation agent generate easy-to-follow sub-instructions considering recognizable and distinctive landmarks based on the agent's visual ability.

2. We construct a high-quality synthetic sub-instruction dataset and design specific tasks for training the translator and the navigation agent.

3. We evaluate our method on R2R, R4R, and R2R-Last, and our method achieves the SOTA results on all benchmarks.

2 Related Work

Vision-and-Language Navigation Anderson et al. (2018) first propose the VLN task with R2R dataset, and many LSTM-based models (Tan et al., 2019; Ma et al., 2019a; Wang et al., 2019; Ma et al., 2019b) show progressing performance. One line of research on this task is to improve the grounding ability by modeling the semantic structure of both the text and vision modalities (Hong et al., 2020a; Li et al., 2021; Zhang and Kordjamshidi, 2022a). Recently, Transformers (Vaswani et al., 2017; Tan and Bansal, 2019; Hong et al., 2021) have been broadly used in the VLN task. VLN↻BERT (Hong et al., 2021) equips a Vision and Language Transformer with a recurrent unit that uses the history information, and HAMT (Chen et al., 2021) has an explicit history learning module and uses Vision Transformer (Dosovitskiy et al., 2020) to learn vision representations. To improve learning representation for the agent, ADAPT (Lin et al., 2022) learns extra prompt features, and CITL (Liang et al., 2022) proposes a contrastive instruction-trajectory learning framework. However, previous works ignore the issue of unrecognizable and nondistinctive landmarks in the instruction, which is detrimental to improving the navigation agent's grounding ability. We propose a translator module that generates easy-to-follow sub-instructions, which helps the agent overcome the abovementioned issues and improves the agent's navigation performance.

Instruction Generation Fried et al. (2018) propose an instruction generator (*e.g.*, Speaker) to generate instructions as the offline augmented data for the navigation agent. Kurita and Cho (2020) design a generative language-grounded policy for the VLN agent to compute the distribution over all possible
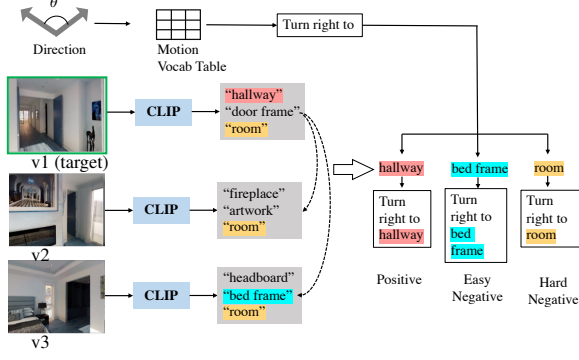
Figure 2: Illustration of constructing the SyFiS dataset.

instructions given action and transition history. Recently, FOAM (Dou and Peng, 2022) uses a bi-level learning framework to model interactions between the navigation agent and the instruction generator. Wang et al. (2022a) propose a cycle-consistent learning scheme that learns both instruction following and generation tasks. In contrast to our work, most prior works rely on the entire trajectory to generate instructions that provide a rather weak supervision signal for each navigation action. Moreover, the previously designed speakers generate textual tokens based on a set of images without considering what instructions are easier for the agent to follow. We address those issues with our designed translator by generating easy-to-follow sub-instruction representations for the navigation agent at each navigation step based on recognizable and distinctive landmarks.

3 Method

In our navigation problem setting, the agent is given an instruction, denoted as $W = \{w_1, w_2, \cdots, w_L\}$, where L is the number of tokens. Also, the agent observes a panoramic view including 36 viewpoints[1] at each navigation step. There are n candidate viewpoints that the agent can navigate to in a panoramic view, denoted as $I = \{I_1, I_2, \cdots, I_n\}$. The task is to generate a trajectory that takes the agent close to a goal destination. The navigation terminates when the navigation agent selects the current viewpoint, or a pre-defined maximum navigation step is reached.

Fig. 3 (a) provides an overall picture of our proposed architecture for the navigation agent. We use VLN↺BERT (Hong et al., 2021) (in Sec. 3.1) as the backbone of our navigation agent and equip it with a novel *translator* module that is trained to convert the full instruction representation into the most relevant sub-instruction representation based

---

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

on the current visual environment. Another key point of our method is to create a synthetic sub-instruction dataset and design the pre-training tasks to encourage the translator to generate effective sub-instruction representations. We describe the details of our method in the following sections.

3.1 Backbone: VLN↺BERT

We use VLN↺BERT as the backbone of our navigation agent. It is a cross-modal Transformer-based navigation agent with a specially designed recurrent state unit. At each navigation step, the agent takes three inputs: text representation, vision representation, and state representation. The text representation X for instruction W is denoted as $X = [x_1, x_2, \cdots, x_L]$. The vision representation V for candidate viewpoints I is denoted as $V = [v_1, v_2, \cdots, v_n]$. The recurrent state representation S_t stores the history information of previous steps and is updated based on X and V_t at the current step. The state representation S_t along with X and V_t are passed to cross-modal transformer layers and self-attention layers to learn the cross-modal representations and select an action, as follows:

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

$$S_{t+1}, a_t = Self\_Attn(\hat{S}_t, \hat{V}_t), \quad (2)$$

we use $\hat{X}, \hat{S}_t, \hat{V}_t$ to represent text, recurrent state, and visual representations after cross-modal transformer layers, respectively. The action is selected based on the self-attention scores between $\hat{S}_t$ and $\hat{V}_t$. S_{t+1} is the updated state representations and a_t contains the probability of the actions.

3.2 Synthetic Sub-instruction Dataset (SyFiS)

This section introduces our novel approach to automatically generate a synthetic fine-grained sub-instruction dataset, SyFiS, which is used to pre-train the *translator* (described in Sec. 3.3) in a contrastive manner. To this aim, for each viewpoint, we generate one positive sub-instruction and three negative sub-instructions. The viewpoints are taken from the R2R dataset (Anderson et al., 2018), and the sub-instructions are generated based on our designed template. Fig. 2 shows an example describing our methodology for constructing the dataset. The detailed statistics of our dataset are included in Sec.4.

The sub-instruction template includes two components: a motion indicator and a landmark. For example, in the sub-instruction "turn left to the
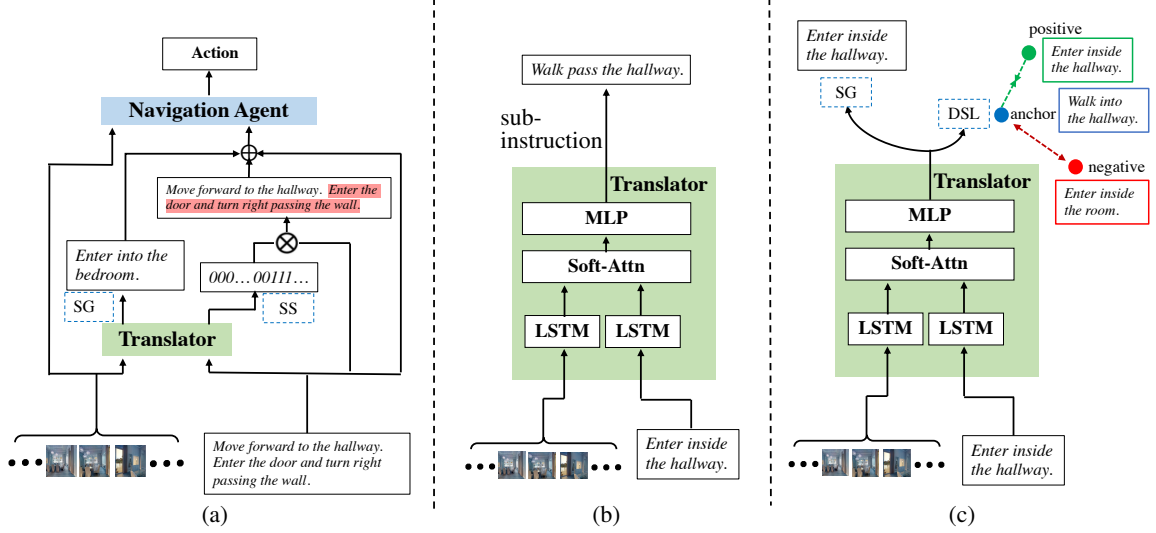
Figure 3: The overview of the proposed method. (a) Navigation agent with VLN-Trans. (b) The translator architecture (c) Pre-training the translator. SG:Sub-instruction Generation; DSL: Distinctive Sub-instruction Learning; SS: Sub-instruction Split.

kitchen", the motion indicator is "turn left", and the landmark is "kitchen". The sub-instruction template is designed based on the semantics of *Spatial Configurations* explained in (Dan et al., 2020).

Motion Indicator Selection First, we generate the motion indicator for the synthesized sub-instructions. Following Zhang et al. (2021), we use pos-tagging information to extract the verbs from instructions in the R2R training dataset and form our motion-indicators dictionary. We divide the motion indicators to 6 categories of: "FOR-WARD", "LEFT", "RIGHT", "UP", "DOWN", and "STOP". Each category has a set of corresponding verb phrases. We refer the Appendix A.1 for more details about motion indicator dictionary.

Given a viewpoint, to select a motion indicator for each sub-instruction, we calculate the differences between the elevation and headings of the current and the target viewpoints. Based on the orientation difference and a threshold, *e.g.* 30 degrees, we decide the motion-indicator category. Then we randomly pick a motion verb from the corresponding category to be used in both generated positive and negative sub-instructions.

Landmark Selection For generating the landmarks for the sub-instructions, we use the candidate viewpoints at each navigation step and select the most *recognizable* and *distinctive* landmarks that are easy for the navigation agent to follow.

In our approach, the most recognizable landmarks are the objects that can be detected by CLIP. Using CLIP (Radford et al., 2021), given a viewpoint image, we predict a label token with the

prompt "a photo of label" from an object label vocabulary. The probability that the image with representation b contains a label c is calculated as follows,

$$p(c) = \frac{exp(sim(b, w_c)/\tau_1)}{\sum_{i=1}^{M}(exp(sim(b, w_i))/\tau_1)}, \quad (3)$$

where τ_1 is the temperature parameter, sim is the cosine similarity between image representation and phrase representation w_c which are generated by CLIP (Radford et al., 2021), M is the vocabulary size. The top-k objects that have the maximum similarity with the image are selected to form the set of recognizable landmarks for each viewpoint.

We filter out the distinctive landmarks from the recognizable landmarks. The distinctive landmarks are the ones that appear in the target viewpoint and not in any other candidate viewpoints. For instance, in the example of Fig. 2, "hallway" is a distinctive landmark because it only appears in the v1 (target viewpoint).

Forming Sub-instructions We use the motion verbs and landmarks to construct sub-instructions based on our template. To form contrastive learning examples, we create positive and negative sub-instructions for each viewpoint. A positive sub-instruction is a sub-instruction that includes a distinctive landmark. The negative sub-instructions include easy negatives and hard negatives. An easy negative sub-instruction contains irrelevant landmarks that appear in any candidate viewpoint except the target viewpoint, *e.g.,* in Fig. 2, "bed frame" appears in v3 and is not observed in the target view-

point. A hard negative sub-instruction includes the nondistinctive landmarks that appear in both the target viewpoint and other candidate viewpoints. For example, in Fig. 2, "room" can be observed in all candidate viewpoints; therefore, it is difficult to distinguish the target from other candidate viewpoints based on this landmark.

3.3 Translator Module

The translator takes a set of candidate viewpoints and the corresponding sub-instruction as the inputs and generates new sub-instructions. The architecture of our translator is shown in Fig. 3(b). This architecture is similar to the LSTM-based Speaker in the previous works (Tan et al., 2019; Fried et al., 2018). However, they generate full instructions from the whole trajectories and use them as offline augmented data for training the navigation agent, while our translator adaptively generates sub-instruction during the agent's navigation process based on its observations at each step.

Formally, we feed text representations of sub-instruction X and the visual representations of candidate viewpoints V into the corresponding LSTM to obtain deeper representation $\tilde{X}$ and $\tilde{V}$. Then, we apply the soft attention between them to obtain the visually attended text representation $\tilde{X}'$, as:

$$\tilde{X}' = SoftAttn(\tilde{X}; \tilde{V}; \tilde{V}) = softmax(\tilde{X}^T W \tilde{V})\tilde{V}, \quad (4)$$

where W is the learned weights. Lastly, we use an MLP layer to generate sub-instruction X' from the hidden representation $\tilde{X}'$, as follows,

$$X' = softmax(MLP(\tilde{X}')) \quad (5)$$

We use the SyFiS dataset to pre-train this translator. We also design two pre-training tasks: Sub-instruction Generation and Distinctive sub-instruction Learning.

Sub-instruction Generation (SG) We first train the translator to generate a sub-instruction, given the positive instructions paired with the viewpoints in the SyfiS dataset as the ground-truth. We apply a cross-entropy loss between the generated sub-instruction X' and the positive sub-instruction X_p. The loss function for the SG task is as follows,

$$L_{SG} = -\frac{1}{L} \sum_L X_p log P(X') \quad (6)$$

Distinctive Sub-instruction Learning (DSL) To encourage the translator to learn sub-instruction

representations that are close to the positive sub-instructions with recognizable and distinctive landmarks, and are far from the negative sub-instructions with irrelevant and nondistinctive landmarks, we use triplet loss to train the translator in a contrastive way. To this aim, we first design triplets of sub-instructions in the form of <anchor, positive, negative>. For each viewpoint, we select one positive and three negative sub-instructions forming three triplets per viewpoint. We obtain the anchor sub-instruction by replacing the motion indicator in the positive sub-instruction with a different motion verb in the same motion indicator category. We denote the text representation of anchor sub-instruction as X_a, positive sub-instruction as X_p, and negative sub-instruction as X_n. Then we feed them to the translator to obtain the corresponding hidden representations $\tilde{X}'_a$, $\tilde{X}'_p$, and $\tilde{X}'_n$ using Eq. 4. The triplet loss function for the DSL task is computed as follows,

$$L_{DSL} = max(D(\tilde{X}'_a, \tilde{X}'_p) - D(X', \tilde{X}'_n) + m, 0), \quad (7)$$

where m is a margin value to keep negative samples far apart, D is the pair-wise distance between representations. In summary, the total objective to pre-train the translator is:

$$L_{pre-train} = \alpha_1 L_{SG} + \alpha_2 L_{DSL} \quad (8)$$

where α_1 and α_2 are hyper-parameters for balancing the importance of the two losses.

3.4 Navigation Agent

We place the pre-trained translator module on top of the backbone navigation agent to perform the navigation task. Fig.3(a) shows the architecture of our navigation agent.

3.4.1 VLN-Trans: VLN with Translator

At each navigation step, the translator takes the given instruction and the current candidate viewpoints as input and generates new sub-instruction representations, which are then used as an additional input to the navigation agent.

Since the given instructions describe the full trajectory, we enable the translator module to focus on the part of the instruction that is in effect at each step. To this aim, we design another MLP layer in the translator to map the hidden states to a scalar attention representation. Then we do the element-wise multiplication between the attention

representation and the instruction representation to obtain the attended instruction representation.

In summary, we first input the text representation of given instruction X and visual representation of candidate viewpoints V to the translator to obtain the translated sub-instruction representation $\tilde{X}'$ using Eq. 4. Then we input $\tilde{X}'$ to another MLP layer to obtain the attention representation X'_m, $X'_m = MLP(\tilde{X}')$. Then we obtain the attended sub-instruction representation as $X'' = X'_m \odot X$, where $\odot$ is the element-wise multiplication.

Lastly, we input text representation X along with translated sub-instruction representation $\tilde{X}'$ and the attended instruction representation X'' into the navigation agent. In such a case, we update the text representation X of VLN$\circlearrowright$BERT as $[X; \tilde{X}'; X'']$, where ; is the concatenation operation.

3.4.2 Training and Inference

We follow (Tan et al., 2019) to train our navigation agent with a mixture of Imitation Learning (IL) and Reinforcement Learning (RL). The IL is to minimize the cross-entropy loss of the predicted and the ground-truth actions. RL is to sample an action from the action probability to learn the rewards. The navigation objective is denoted as:

$$L_{nav} = -\sum_t -a_t^s log(p_t^a) - \lambda \sum_t a_t^* log(p_t^a) \quad (9)$$

where a_t^s is the sampled action for RL, a_t^* is the teacher action, and λ is the coefficient.

During the navigation process, we design two auxiliary tasks specific to the translator. The first task is still the SG task in pre-training to generate the correct sub-instructions; the second task is Sub-instruction Split (SS), which generates the correct attended sub-instruction. Specifically, for the SS task, at each step, we obtain the ground-truth attention representation by labeling the tokens of the sub-instruction in the full instruction as 1 and other tokens as 0. We denote ground-truth attended sub-instruction representation as X_m. Then, we apply Binary Cross Entropy loss between X_m and the generated attention representation X'_m as follows,

$$L_{SS} = -\frac{1}{L} \sum_L X_m log(X'_m) \quad (10)$$

The overall training objective of the navigation agent including the translator's auxiliary tasks is:

$$L_{obj} = \beta_1 L_{nav} + \beta_2 L_{SG} + \beta_3 L_{SS}, \quad (11)$$

where β_1, β_1, and β_3 are the coefficients. During inference, we use the greedy search to select an action with the highest probability at each step to finally generate a trajectory.

4 Experiments

4.1 Dataset and Evaluation Metrics

Dataset We evaluate our approach on three datasets: **R2R** (Anderson et al., 2018), **R4R** (Jain et al., 2019), and **R2R-Last** (Chen et al., 2021). R2R includes $21,567$ instructions and $7,198$ paths. The entire dataset is partitioned into training, seen validation and unseen validation, and unseen test sets. R4R extends R2R with longer instructions by concatenating two adjacent tail-to-head trajectories in R2R. R2R-Last uses the last sentence in the original R2R to describe the final destination instead of step-by-step instructions.

Evaluation Metrics Three metrics are used for navigation (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) that normalizes the success rate with trajectory length. The R4R dataset uses two more metrics to measure the fidelity between the predicted and the ground-truth path: (4) Coverage Weighted by Length Score (CLS) (Jain et al., 2019). (5) Normalized Dynamic Time Warping weighted by Success Rate (sDTW) (Ilharco et al., 2019). We provide a more detailed description of the dataset and metrics in the Appendix A.3.

4.2 Implementation Details

We use ResNet-152 (He et al., 2016) pre-trained on Places365 (Zhou et al., 2017) as the visual feature and the pre-trained BERT (Vaswani et al., 2017) representation as the initialized text feature. We first pre-train the translator and navigation agent offline. Then we include the translator in the navigation agent to train together. To pre-train the translator, we use one NVIDIA RTX GPU. The batch size and learning rate are 16 and $1e-5$, respectively. Both α_1 and α_2 in Eq. 8 are 1. To pre-train the navigation agent, we follow the methods in Zhang and Kordjamshidi (2022b) and use extra pre-training datasets to improve the baseline. We use 4 GeForce RTX 2080 GPUs(~2 days), and the batch size on each GPU is 28.The learning rate

		Val seen			Val Unseen			Test Unseen		
	Method	NE ↓	SR ↑	SPL↑	NE ↓	SR ↑	SPL↑	NE ↓	SR ↑	SPL ↑
1	Env-Drop (Tan et al., 2019)	3.99	0.62	0.59	5.22	0.47	0.43	5.23	0.51	0.47
2	RelGraph (Hong et al., 2020a)	3.47	0.67	0.65	4.73	0.57	0.53	4.75	0.55	0.52
3	NvEM (An et al., 2021)	3.44	0.69	0.65	4.27	0.60	0.55	4.37	0.58	0.54
4	PREVALENT (Hao et al., 2020)	3.67	0.69	0.65	4.71	0.58	0.53	5.30	0.54	0.51
5	HAMT (ResNet) (Chen et al., 2021)	−	0.69	0.65	−	0.64	0.58	−	−	−
6	HAMT (ViT) (Chen et al., 2021)	2.51	0.76	0.72	−	0.66	0.61	**3.93**	0.65	**0.60**
7	CITL (Liang et al., 2022)	2.65	0.75	0.70	3.87	0.63	0.58	3.94	0.64	0.59
8	ADAPT (Lin et al., 2022)	2.70	0.74	0.69	3.66	0.66	0.59	4.11	0.63	0.57
9	LOViS (Zhang and Kordjamshidi, 2022b)	**2.40**	0.77	0.72	3.71	0.65	0.59	4.07	0.63	0.58
10	VLN↻BERT (Hong et al., 2021)	2.90	0.72	0.68	3.93	0.63	0.57	4.09	0.63	0.57
11	VLN↻BERT$^+$ (ours)	2.72	0.75	0.70	3.65	0.65	0.60	4.09	0.63	0.57
12	VLN↻BERT^{++} (ours)	2.51	0.77	0.72	3.40	0.67	0.61	4.02	0.63	0.58
13	VLN-Trans-R2R (ours)	**2.40**	**0.78**	**0.73**	3.37	0.67	**0.63**	3.94	0.65	0.59
14	VLN-Trans-FG-R2R (ours)	2.45	0.77	0.72	**3.34**	**0.69**	**0.63**	3.94	**0.66**	**0.60**

Table 1: Experimental results on R2R Benchmarks in a single-run setting. The best results are in bold font. + means we add RXR (Ku et al., 2020) and Marky-mT5 dataset (Wang et al., 2022b) as the extra data to pre-train the navigation agent. ++ means we further add SyFiS dataset to pre-train the navigation agent. ViT means Vision Transformer representations.

		Val Seen					Val Unseen				
	Method	NE↑	SR↑	SPL↑	CLS↑	sDTW↑	NE↓	SR↑	SPL↑	CLS↑	sDTW↑
1	OAAM (Qi et al., 2020a)	-	0.56	0.49	0.54	-	0.32	0.29	0.18	0.34	0.11
2	RelGraph (Hong et al., 2020a)	5.14	0.55	0.50	0.51	0.35	7.55	0.35	0.25	0.37	0.18
3	NvEM (An et al., 2021)	5.38	0.54	0.47	0.51	0.35	6.80	0.38	0.28	0.41	0.20
4	VLN↻BERT* (Hong et al., 2021)	4.82	0.56	0.46	0.56	0.38	6.48	0.43	0.32	0.42	0.21
5	CITL (Liang et al., 2022)	**3.48**	0.67	0.57	0.56	**0.43**	6.42	0.44	0.35	0.39	0.23
6	LOViS (Zhang and Kordjamshidi, 2022b)	4.16	**0.67**	0.58	**0.58**	**0.43**	6.07	0.45	0.35	**0.45**	0.23
7	VLN-Trans	3.79	**0.67**	**0.59**	0.57	**0.43**	**5.87**	**0.46**	**0.36**	**0.45**	**0.25**

Table 2: Experimental results on R4R dataset in a single-run setting. * denotes our reproduced R4R results.

	Val Seen		Val Unseen	
Method	SR↑	SPL↑	SR↑	SPL↑
EnvDrop (Tan et al., 2019)	0.43	0.38	0.34	0.28
VLN↻BERT (Hong et al., 2020a)	0.50	0.46	0.42	0.37
HAMT (Chen et al., 2021)	0.53	0.50	0.45	0.41
VLN-Trans	**0.58**	**0.53**	**0.50**	**0.45**

Table 3: Experimental results on the R2R-Last dataset.

is $5e − 5$.

We further train the navigation agent with a translator for 300K iterations using an NVIDIA RTX GPU (~1 day). The batch size is 16, and the learning rate is $1e − 5$. The optimizer is AdamW (Loshchilov and Hutter, 2017). We can get the best results when we set $λ$ as 0.2 in Eq. 9 , and $β_1$, $β_2$, and $β_3$ as 1, 1 and 0.1 in Eq. 11, respectively. The best model is selected according to performance on val unseen split. Please check our code [2] for the implementation.

4.3 Experimental Results

Table 1 shows the model performance on the R2R benchmarks. Row #4 to row #9 are Transformer-based navigation agents with pre-trained cross-modality representations, and such representations greatly improve performance of LSTM-based VLN models (row #1 to row #3). It is impressive that our VLN-Trans model's performance (row #13 and row

#14) on both validation seen and unseen performs 2%-3% better than HAMT (Chen et al., 2021) when it even uses more advanced ViT (Dosovitskiy et al., 2020) visual representations compared with ResNet. Our performance on both SR and SPL are still 3%-4% better than the VLN agent using contrastive learning: CITL (Liang et al., 2022) (row #7) and ADAPT (Lin et al., 2022) (row #8). LOViS (Zhang and Kordjamshidi, 2022b) (row #9) is another very recent SOTA improving the pretraining representations of the navigation agent, but we can significantly surpass their performance. Lastly, compared to the baseline (row #10), we first significantly improve the performance (row #11) by using extra augmented data, Room-across-Room dataset (RXR) (Ku et al., 2020) and the Marky-mT5 (Wang et al., 2022b), in the pre-training of navigation agent. The performance continues to improve when we further include the SyFiS dataset in the pre-training, as shown in row #12, proving the effectiveness of our synthetic data. Row #13 and row #14 are the experimental results after incorporating our pre-trained translator into the navigation model. First, for a fair comparison with other models, we follow the baseline (Hong et al., 2021) to train the navigation agent using the R2R (Anderson et al., 2018) dataset and the augmented data from PREVALENT (Hao et al.,

2020). Since those datasets only contain the pairs of full instructions and the trajectories without intermediate alignments between sub-instructions and the corresponding viewpoints, we do not optimize the translator ($\beta2 = 0$, $\beta3 = 0$ in Eq.11) during training the navigation agent, which is denoted as VLN-Trans-R2R. As shown in row #13, our translator helps the navigation agent obtain the best results on the seen environment and improves SPL by 2% on the unseen validation environment, proving that the generated sub-instruction representation enhances the model's generalizability. However, FG-R2R (Hong et al., 2020b) provides human-annotated alignments between sub-instructions and viewpoints for the R2R dataset, and our SyFiS dataset also provides synthetic sub-instructions for each viewpoint. Then we conduct another experiment using FG-R2R and SyFiS datasets to train the navigation agent. Simultaneously, we optimize the translator using the alignment information with our designed SG and SS losses during the navigation process. As shown in row #13, we further improve the SR and SPL on the unseen validation environment. This result indicates our designed losses can better utilize the alignment information.

Table 2 shows results on the R4R benchmark. Row #1 to Row #3 are the LSTM-based navigation agent. Row #4 reports our re-implemented results of VLN↻BERT, and both CITL and LOViS are the SOTA models. Our method (row #7) improves the performance on almost all evaluation metrics, especially in the unseen environment. The high sDTW means that our method helps navigation agents reach the destination with a higher successful rate and better follow the instruction.

Table 3 shows the performance on the R2R-Last benchmark. When only the last sub-sentence is available, our translator can generate a sub-instruction representation that assists the agent in approaching the destination. As shown in Table 3, we improve the SOTA (Row #3) by almost 5% on the SR in the unseen validation dataset. We obtain the best results on R2R-Last without the Sub-instruction Split task. More details are in the ablation study (see Sec. 4.4).

4.4 Ablation Study

In Table 4, we show the performance after ablating different tasks in the baseline model on the R2R and R2R-Last datasets. We compared with VLN↻BERT[++], which is our improved baseline

Dataset	Method	Tasks			Val Seen		Val Unseen	
		SG	DSL	SS	SR↑	SPL↑	SR↑	SPL↑
R2R	Baseline				0.767	0.722	0.672	0.611
	1	✔			0.764	0.721	0.673	0.623
	2	✔	✔		**0.780**	**0.728**	0.674	0.627
	3	✔	✔	✔	0.772	0.720	**0.690**	**0.633**
R2R-Last	Baseline				0.552	0.501	0.473	0.422
	1	✔			0.573	0.521	0.494	0.434
	2	✔	✔		**0.582**	**0.534**	**0.503**	**0.453**
	3	✔	✔	✔	0.571	0.511	0.484	0.433

Table 4: Ablation study, where Baseline is VLN↻BERT[++].

after adding extra pre-training data to the navigation agent. First, we pre-train our translator with SG and DSL tasks and incorporate the translator into the navigation agent without further training. For both the R2R dataset and R2R-Last, SG and DSL pre-training tasks can incrementally improve the unseen performance (as shown in method 1 and method 2 for R2R and R2R-Last). Then we evaluate the effectiveness of the SS task when we use it to train the translator together with the navigation agent. For the R2R dataset, the model obtains the best result on the unseen environment after using the SS task. However, the SS task causes the performance drop for the R2R-Last dataset. This is because the R2R-Last dataset merely has the last single sub-instruction in each example and there is no other sub-instructions our model can identify and learn from.

4.5 Qualitative Study

Statistic of the SyFiS dataset We construct SyFiS dataset using $1,076,818$ trajectories, where 7198 trajectories are from the R2R dataset, and $1,069,620$ trajectories are from the augmented data (Hao et al., 2020). Then we pair those trajectories with our synthetic instructions to construct the SyFiS dataset based on our pre-defined motion verb vocabulary and CLIP-generated landmarks (in Sec3.2). When we pre-train the translator, we use the sub-instruction of each viewpoint in a trajectory. There are usually 5 to 7 viewpoints in a trajectory; each viewpoint is with one positive sub-instruction and three negative sub-instructions.

Quality of SyFiS dataset. We randomly select 50 instructions from the SyFiS dataset and manually check if humans can easily follow those instructions. As a result, we achieve 58% success rate. It is reported (Wang et al., 2022b) that success rate of the generated instruction are 38% and 48% in Speaker-Follower (Fried et al., 2018) and Env-dropout (Tan et al., 2019), respectively. The 10% higher success rate of our instructions indicates

we have synthesized a better quality dataset for pre-training and fine-tuning.

Translator Analysis Our translator can relate the mentioned landmarks in the instruction to the visible and distinctive landmarks in the visual environment. In Fig. 4 (a), "tables" and "chairs" are not visible in three candidate viewpoints (v1-v3). However, our navigation agent can correctly recognize the target viewpoint using the implicit instruction representations generated by the translator. We assume the most recognizable and distinctive landmark, that is, the "patio" here in the viewpoint v3 has a higher chance to be connected to a "table" and a "chair" based on our pre-training, compared to the landmarks in the other viewpoints. In Fig. 4 (b), both candidate viewpoints v2 and v3 contain kitchen (green bounding boxes); hence it is hard to distinguish the target between them. However, for the translator, the most distinctive landmark in v3 is the "cupboard" which is more likely to be related to the "kitchen". Fig. 4(c) shows a failure case, in which the most distinctive landmark in candidate viewpoint v1 is "oven". It is more likely for the translator relates "oven" to the "kitchen" compared to "countertop", and the agent selects the wrong viewpoints. In fact, we observe that the R2R validation unseen dataset has around 300 instructions containing "kitchen". For corresponding viewpoints paired with such instructions, our SyFiS dataset generates 23 and 5 sub-instructions containing "oven" and "countertop", respectively, indicating the trained translator more likely relates "oven" to "kitchen". More examples are shown in Appendix. A.4.

5   Conclusion

In the VLN task, instructions given to the agent often include landmarks that are not recognizable to the agent or are not distinctive enough to specify the target. Our novel idea to solve these issues is to include a translator module in the navigation agent that converts the given instruction representations into effective sub-instruction representations at each navigation step. To train the translator, we construct a synthetic dataset and design pre-training tasks to encourage the translator to generate the sub-instruction with the most recognizable and distinctive landmarks. Our method achieves the SOTA results on multiple navigation datasets. We also provide a comprehensive analysis to show the effectiveness of our method. It is worth noting that while we focus on R2R, the novel components

(a) Walk left past the table and chairs and through the doorway.

(b) Walk towards the kitchen area. Keep walking along the kitchen area towards the doorway.

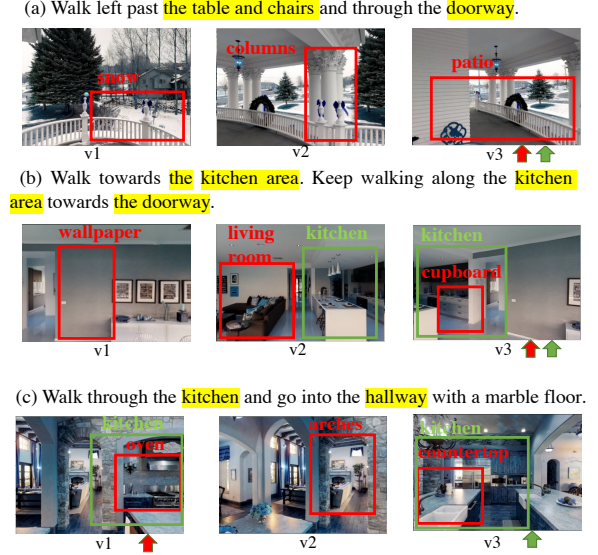(c) Walk through the kitchen and go into the hallway with a marble floor.

Figure 4: Qualitative examples to show how the translator helps the navigation agent. The red boxes and green boxes show the distinctive and the nondistinctive landmarks; the green arrow and red arrow show the target and the predicted viewpoints.

of our technique for generating synthetic data and pre-training the translator are easily applicable to other simulation environments.

6   Limitations

We mainly summarize three limitations of our work. First, the translator only generates a representation, not an actual instruction, making the model less interpretable. Second, we do not include more advanced vision representations such as ViT and CLIP to train the navigation agent. Although only using ResNet, we already surpass prior methods using those visual representations (e.g., HAMT (Chen et al., 2021)), it would be interesting to experiment with those different visual representations. Third, this navigation agent is trained in a simulated environment, and a more realistic setting will be more challenging.

7   Acknowledgement

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.

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.

Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. 2017. Matterport3d: Learning from rgb-d data in indoor environments. *arXiv preprint arXiv:1709.06158*.

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.

Soham Dan, Parisa Kordjamshidi, Julia Bonn, Archna Bhatia, Zheng Cai, Martha Palmer, and Dan Roth. 2020. From spatial relations to spatial configurations. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 5855–5864.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Zi-Yi Dou and Nanyun Peng. 2022. Foam: A follower-aware speaker model for vision-and-language navigation. *arXiv preprint arXiv:2206.04294*.

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.

Jing Gu, Eliana Stefani, Qi Wu, Jesse Thomason, and Xin Eric Wang. 2022. Vision-and-language navigation: A survey of tasks, methods, and future directions. *arXiv preprint arXiv:2203.12667*.

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, Qi Wu, and Stephen Gould. 2020b. Sub-instruction aware vision-and-language navigation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3360–3376.

Yicong Hong, Qi Wu, Yuankai Qi, Cristian Rodriguez-Opazo, and Stephen Gould. 2021. Vln bert: A recurrent vision-and-language bert for navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1643–1653.

Ronghang Hu, Daniel Fried, Anna Rohrbach, Dan Klein, Trevor Darrell, and Kate Saenko. 2019. Are you looking? grounding to multiple modalities in vision-and-language navigation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6551–6557.

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*.

Shuhei Kurita and Kyunghyun Cho. 2020. Generative language-grounded policy in vision-and-language navigation with bayes' rule. *arXiv preprint arXiv:2009.07783*.

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

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.

Bingqian Lin, Yi Zhu, Zicong Chen, Xiwen Liang, Jianzhuang Liu, and Xiaodan Liang. 2022. Adapt: Vision-language navigation with modality-aligned action prompts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15396–15406.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

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

Chih-Yao Ma, Zuxuan Wu, Ghassan AlRegib, Caiming Xiong, and Zsolt Kira. 2019b. The regretful agent: Heuristic-aided navigation through progress estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6732–6740.

Yuankai Qi, Zizheng Pan, Shengping Zhang, Anton van den Hengel, and Qi Wu. 2020a. Object-and-action aware model for visual language navigation. In *European Conference on Computer Vision*, pages 303–317. Springer.

Yuankai Qi, Qi Wu, Peter Anderson, Xin Wang, William Yang Wang, Chunhua Shen, and Anton van den Hengel. 2020b. Reverie: Remote embodied visual referring expression in real indoor environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9982–9991.

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.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28.

Hao Tan and Mohit Bansal. 2019. Lxmert: Learning cross-modality encoder representations from transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5100–5111.

Hao Tan, Licheng Yu, and Mohit Bansal. 2019. Learning to navigate unseen environments: Back translation with environmental dropout. In *Proceedings of NAACL-HLT*, pages 2610–2621.

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, Wei Liang, Jianbing Shen, Luc Van Gool, and Wenguan Wang. 2022a. Counterfactual cycle-consistent learning for instruction following and generation in vision-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15471–15481.

Su Wang, Ceslee Montgomery, Jordi Orbay, Vighnesh Birodkar, Aleksandra Faust, Izzeddin Gur, Natasha Jaques, Austin Waters, Jason Baldridge, and Peter Anderson. 2022b. 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.

Xin Wang, Qiuyuan Huang, Asli Celikyilmaz, Jianfeng Gao, Dinghan Shen, Yuan-Fang Wang, William Yang Wang, and Lei Zhang. 2019. Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6629–6638.

Wansen Wu, Tao Chang, and Xinmeng Li. 2021. Vision-language navigation: A survey and taxonomy. *arXiv preprint arXiv:2108.11544*.

Yue Zhang, Quan Guo, and Parisa Kordjamshidi. 2021. Towards navigation by reasoning over spatial configurations. *SpLU-RoboNLP 2021*, page 42.

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. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 5745–5754.

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.

Motions	Vocab
FORWARD	go forward to; go forward past; pass; walk pass, walk forward, etc.
DOWN	go down; walk straight down; walk down; move forward down, etc.
UP	go up; walk up; climb; leading upwards; travel up, etc.
RIGHT	turn right to; make a right turn to, go right to, veer right to, etc.
LEFT	walk left to, turn left to, go left to, make a left turn to, make a left to, etc.
STOP	stay at, stand by, stop by, wait by, stop at, wait on, etc.

Figure 5: Motion Indicator Vocabulary

A Appendix

A.1 Motion Indicator Dictionary

We extract the motion verb phrases in the R2R training instructions to build a motion indicator dictionary, as shown in Fig. 5. We first use spaCy [3] to extract motion verbs based on pos-tagging information , and then manually collect the prepositions after the motion verbs, such as "stop at", " stop by" and " stop behind of". In summary, there are 131 verb phrases for the action of "FORWARD", 11 verb phrases for the action of "DOWN", 11 verb phrases for the action of"UP", 28 verb phrases for the action of"LEFT", 23 for the action of "RIGHT", and 26 for the action of "STOP".

A.2 Comparison among different datasets

One of the contributions of our method is the proposed SyFiS dataset, which forms sub-instruction for each viewpoint considering recognizable and distinguishable landmarks. In this section, we compare different datasets to show the main improvements of the SyFiS compared to other datasets. As shown in Fig. 6, in the R2R dataset (Anderson et al., 2018), instructions describe the entire trajectory, which is challenging for the navigation agent to follow in every single step. Based on it, FG-R2R (Hong et al., 2020b) provides a manual annotation to align the sub-instruction to the corresponding viewpoints. Although providing fine-grained annotation, the sub-instructions in FG-R2R are still not step-by-step. ADAPT (Lin et al., 2022) generates the sub-instruction for every single viewpoint. However, they only consider the viewpoints in trajectory and select the most obvious landmarks for each target viewpoint. Those selected landmarks are quite general, and hard to distinguish the target viewpoint from other candidate viewpoints, such as the "living room", "hallway" and "bedroom". Nevertheless, both FG-R2R and ADAPT still suffer from the issue of nondistinctive landmarks, such as the "living room", "hallway" and "bedroom", which hurts the navigation performance, as stated

R2R: v1 → v2 → v3 → v4 → v5
Turn right and head past the kitchen down the long hallway ahead. Go past the bathroom and stop just inside the bedroom.

FG-R2R: v1 → v2 → v3 Turn right and head past the kitchen down the long hallway ahead.
v4 Go past the bathroom.
v5 Stop just inside the bedroom.

ADAPT:
v1 → v2 → v3 → v4 → v5

v1 Walk out of the living room.
v2 Go into the hallway.
v3 Go into the hallway.
v4 Go to the bedroom.
v5 Stop at the bedroom.

SyFiS:

v1 Turn right to the entertainment center.
v2 Move towards the picture.
v3 Walk towards the hallway.
v4 Go all the way towards the window.
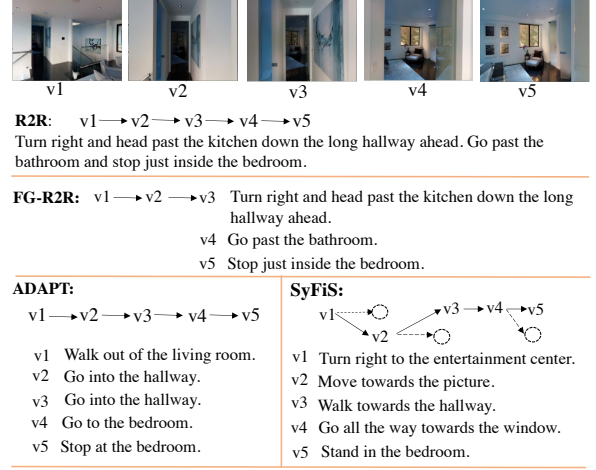v5 Stand in the bedroom.

Figure 6: Comparisons among Different Datasets.

previously. We construct a dataset with the most recognizable and distinguishable landmark, which is obtained by comparing the target viewpoint with other candidate viewpoints at each navigation step. Based on our experimental results, our generated sub-instruction dataset can largely help the navigation performance.

A.3 Evaluation Datasets and Metrics

Our method is evaluated on R2R (Anderson et al., 2018), R4R (Jain et al., 2019), and R2R-Last (Chen et al., 2021). All these three dataset are built upon the Matterport3D (Chang et al., 2017) indoor scene dataset.

R2R provides long instructions paired with the corresponding trajectory. The dataset contains 61 houses from training, 56 houses for validation in seen environment, 11 and 18 houses for unseen environment validation and test, respectively. The seen set shares the same visual environment with training dataset, while unseen sets contain different environments.

R4R extends R2R by concatenating two trajectories and their corresponding instructions. In R4R, trajectories are less biased compared to R2R, because they are not necessarily the shortest path from the source viewpoint to the target viewpoint.

R2R-Last proposes a VLN setup that is similar to that of REVERIE (Qi et al., 2020b), which only claims the destination position. More formally, R2R-Last only leverages the the last sentence in the original R2R instructions to describe the final destination.

Evaluation Metrics VLN task mainly evaluates navigation agent's generalizability in unseen

Instruction: Walk towards fireplace.

v1   v2   v3

(a) The landmark of the "fireplace" in the given instruction can not be observed in all three candidate viewpoints. The target viewpoint v3 contains a distinctive landmark "living room", and the translator can relate it to "fireplace", which helps the agent select the correct target viewpoint.

Instruction: Walk though the kitchen. Stop on the balcony.

v1   v2   v3

(b) The target viewpoint v1 contains landmarks of "apartment", "cupboard" and "skylight". Among them, "apartment" and "cupboard" are nondistinctive because they appear in both target and other candidate viewpoints. Our agent can select the correct targe viewpoint because our translator can relate "skylight" to "balcony". According to our observations, in the R2R training data, among the trajectories paired with the instructions containing "balcony", our SyFiS dataset generates 14% sub-instructions containing "skylight" for the corresponding viewpoints.

Instruction: Go down the hallway and enter into the bathroom.

v1   v2   v3   v4

(c) The landmarks of the "bathroom" in the instruction can not be observed in all viewpoints. The target viewpoint v1 contains landmarks of "hallway" and "vanity", where "hallway" is nondistinctive since it also can be observed in other candidate viewpoints. Our translator relates "vanity" to "bathroom", which helps the agent select the correct viewpoint.

Instruction: Turn right and go down the hall.

v1   v2   v3   v4

(d) In this case, all candidate viewpoints include "hall". According to our observation, in the R2R training data, among the trajectories paired with the instructions containing "hall", our SyFiS dataset generates 3% sub-instructions containing "pillar", 1% containing "court", and almost 0% containing "church".In such a case, our translator has a higher chance of relating "hall" to "pillar" and selecting the wrong viewpoint.

Instruction: Exit the kitchen, and walk down the hall.

v1   v2

(e) Both viewpoints contain "kitchen" and "hall", but our translator highly relates "kitchen" to "cabinets" compared to "oven". In the R2R training data, among the trajectories paired with the instructions containing "kitchen", our SyFiS dataset generates 9% sub-instructions containing "cabinet" while 6% containing "oven". In such a case, our translator is more likely to relate "kitchen" to "cabinets" and select the wrong viewpoint.

Figure 7: Qualitative Examples. (a)(b)(c) are correct examples, and (d)(e) are wrong examples. The red boxes and green boxes show the distinctive and nondistinctive landmarks based on the target viewpoint; The green arrow and red arrow show the target and the predicted viewpoint from model.

environment using validation unseen and test unseen datasets. Success Rate (SR) and Success Rate weighted Path length (SPL) are two main metrics for all three datasets, where a predicted path is *success* if the agent stop within 3 meters of the destination. The metrics of SR and SPL can evaluate the accuracy and efficiency of navigation.

A.4 Qualitative Examples for Translator Analysis

We provide more qualitative examples in Fig. 7 to show our translator can relate the mentioned landmarks in the instruction to the recognizable and distinctive landmarks in the visual environment. Fig. 7(a)(b)(c) shows successful cases in that our translator helps the navigation agent make correct decisions. However, there are chances our translator relates to wrong landmarks in the visual environment because of biased data. This may lead to the wrong decisions of the navigation agent, and we provide failure cases in Fig. 7(e)(f).