Sub-Instruction Aware Vision-and-Language Navigation

Yicong Hong[*1], **Cristian Rodriguez-Opazo**[*1], **Qi Wu**[2], **Stephen Gould**[1]
[1]The Australian National University, [2]University of Adelaide
[1,2]Australian Centre for Robotic Vision
{yicong.hong, cristian.rodriguez, stephen.gould}@anu.edu.au
qi.wu01@adelaide.edu.au

Abstract

Vision-and-language navigation requires an agent to navigate through a real 3D environment following natural language instructions. Despite significant advances, few previous works are able to fully utilize the strong correspondence between the visual and textual sequences. Meanwhile, due to the lack of intermediate supervision, the agent's performance at following each part of the instruction cannot be assessed during navigation. In this work, we focus on the granularity of the visual and language sequences as well as the traceability of agents through the completion of an instruction. We provide agents with fine-grained annotations during training and find that they are able to follow the instruction better and have a higher chance of reaching the target at test time. We enrich the benchmark dataset Room-to-Room (R2R) with sub-instructions and their corresponding paths. To make use of this data, we propose effective sub-instruction attention and shifting modules that select and attend to a single sub-instruction at each time-step. We implement our sub-instruction modules in four state-of-the-art agents, compare with their baseline models, and show that our proposed method improves the performance of all four agents.

We release the Fine-Grained R2R dataset (FGR2R) and the code at https://github.com/YicongHong/Fine-Grained-R2R.

1 Introduction

Creating an agent that can navigate through an unknown environment following natural language instructions has been a dream of human-beings for many years. Such an agent needs to possess the ability to perceive its environment, understand the instructions and learn the relationship between these two streams of information. Recently, Anderson et al. (2018b) proposed the vision-and-language navigation (VLN) task that formalized such requirements through an evaluation of an agent's ability to follow natural language instructions in photo-realistic environments.

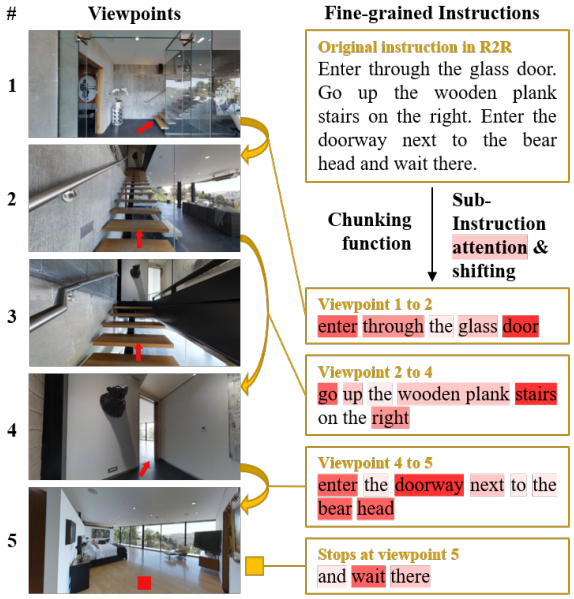


Figure 1: Visual navigation with sub-instruction and sub-path pairs. We enrich the R2R dataset by providing fine-grained matching between sub-instructions and viewpoints along the ground-truth path.

Despite the significant progress made by recent approaches, there is little evidence that agents learn the correspondence between observations and instructions. Hu et al. (2019) found that a modified self-monitoring agent (Ma et al., 2019a), could achieve similar performance with (success rate 40.5%) and without (success rate 39.7%) visual information. Among other reasons, such as dataset bias, the result suggests that this agent gains little from having the two streams of information.

We argue that one of the main reasons be-

* Authors contributed equally

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 3360–3376,
November 16–20, 2020.

hind this is that current methods are not adequately teaching agents the relationship between perception—things that the robot is observing—and parts of the instructions. Since datasets do not provide such information agents can only use the ground-truth trajectory as a learning signal. Moreover, given the lack of fine-grained annotation, current methods cannot evaluate the (perceptual or linguistic) grounding process at each step as there is no ground truth signal to indicate which part of the instruction has been completed.

To address this problem, we enhance the R2R dataset (Anderson et al., 2018a) to acquire intermediate supervision for the agents, providing a fine-grained matching between sub-instructions and the agent's visual perception, as illustrated in Figure 1, to produce our Fine-Grained Room-to-Room dataset (FGR2R). We argue that the granularity of the navigation task should be at the level of these sub-instructions, rather than attempting to ground a specific part of the original long and complex instruction without any direct supervision or measure navigation progress at word level.

Our work aims to make the navigation process traceable and encourage the agent to run precisely on the described path rather than just focusing on reaching the target. We hypothesize that the agent should reach the target with higher success rate by following a detailed instruction with richer information, and in practice, the agent could complete some additional tasks on its way to the target.

In light of this, we propose a novel sub-instruction attention mechanism to better learn the correspondence between visual features and language features. Our agent first segments the long and complicated instruction into short and easier-to-understand sub-instructions using a heuristic method based on the grammatical relations provided by the Stanford NLP Parser (Qi et al., 2018). Moreover, we propose a shifting module that infers whether the current sub-instruction has been completed. Hence, only one sub-instruction is available to the agent at each time step for textual grounding. These modules can be easily applied to previous VLN models.

We conduct experiments to compare the performance of four state-of-the-art agents to evaluate with or without our sub-instruction module, for agents based on imitation learning (Anderson et al., 2018b; Fried et al., 2018; Ma et al., 2019a) and reinforcement learning (Tan et al., 2019). Analyzing the results we find that the intermediate supervision and our proposed modules help the agents to better follow the instructions. Furthermore, we demonstrate the traceability of the navigation process through qualitative and quantitative analysis.

2 Related Work

Visual and textual grounding. Visual grounding aims to infer the relationship between a text description and a spatial or temporal region in an image or video, respectively. It is an essential component for a variety of tasks in vision-and-language research such as visual question answering (VQA) (Schwartz et al., 2017; Anderson et al., 2018a; Hudson and Manning, 2019), image captioning (Xu et al., 2015; Anderson et al., 2018a; Cornia et al., 2019; Ma et al., 2020), video understanding (Gao et al., 2017; Ma et al., 2018; Rodriguez et al., 2020) and phrase localization (Engilberge et al., 2018; Yu et al., 2018). In the case of vision-and-language navigation, at each navigational step, the agent attends to the relevant part of the instruction according to visual clues to direct the future action. Meanwhile, the agent attends the visual inputs at different directions as described by text to perceive the environment (Fried et al., 2018; Ma et al., 2019a).

Vision and language navigation. Anderson et al. (2018b) formalized the vision-and-language navigation task in a photo-realistic environment, and proposed a benchmark Room-to-Room (R2R) dataset and a sequence-to-sequence agent as a baseline model. Other datasets in real environments, such as R4R (Jain et al., 2019), which is an extended version of R2R with longer instruction-path pairs, and Touchdown (Chen et al., 2019) for navigation on streets have also been proposed for study.

Researchers have addressed the R2R task through a great variety of approaches. Wang et al. (2018) propose a look-ahead model that combines model-based and model-free reinforcement learning, predicts the agent's next state and reward during navigation. Fried et al. (2018) proposed the Speaker-Follower model which generates augmented samples for training and makes use of the panoramic action space to ground and navigate efficiently. Later, Ma et al. (2019a) introduced the Self-Monitoring agent which includes a vision and language co-grounding network and a progress monitor. The progress monitor estimates a normalized distance to the target and guides the transition of the textual attention. Wang et al. (2019) applied

the REINFORCE algorithm (Williams, 1992) to improve the agent's generalizability and proposed a Self-Supervised Imitation Learning (SIL) method to facilitate lifelong learning in a new environment. The Back Translation agent (Tan et al., 2019) applied the A2C algorithm (Mnih et al., 2016) and made use of a speaker module with environmental dropout for data augmentation. Landi et al. (2019) applied dynamic convolutional filters for image feature extraction for low-level grounding of visual inputs and Hu et al. (2019) grounded multiple modalities using a mixture-of-experts approach and applied joint training strategy. Besides, the Regretful agent (Ma et al., 2019b) and the Tactical Rewind agent (Ke et al., 2019) are models which focus on path-scoring and backtracking methods. Very recently, Zhu et al. (2020a) introduces multiple auxiliary losses in training to help exploring the semantic meaning of visual features, Huang et al. (2019) and Hao et al. (2020) apply pre-trained encoders to generate generic visual and textual representations for the agent.

In contrast to all previously mentioned methods that ground the complete instruction, we propose to divide the instruction into meaningful semantic sub-instructions, and teach the agent to complete each one at a time before reading the next sub-instruction. In that spirit, our method is similar to the image captioning work by Cornia et al. (2019). They design a shifting gate over the image regions to control the visual features that feed into each time step of the caption module. We differ from their work in the modality that is attended. Our method works in the language domain, and the shifting depends only on local context rather than looking over all the sub-instructions. BabyWalk (Zhu et al., 2020b) is a concurrent work to ours, it uses sub-instructions for curriculum learning which trains the agent to complete shorter navigation tasks before trying to solve the longer ones. Comparing the sub-instruction and sub-path pairs in FGR2R and BabyWalk, BabyWalk aligns the textual and visual sequences by solving a dynamic programming problem, whereas FGR2R employs human annotation, which is more fine-grained and accurate.

3 Sub-instruction Aware VLN

In this section, we first introduce the VLN problem and the general architecture of the agent. Then, we discuss about the proposed chunking function for producing the sub-instructions and the novel sub-instruction module for enabling sub-instruction attention and transition.

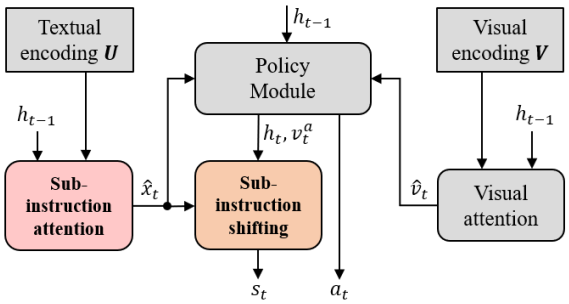


Figure 2: Our sub-instruction attention and shifting modules built into the self-monitoring agent pipeline. We replace the original textual attention module with our sub-instruction modules that select and attend a single sub-instruction at each time-step.

The VLN task requires the agent to navigate through a real environment to a target location following a natural language instruction. Formally, an instruction $\boldsymbol{w}$ is a sequence of words $\langle w_1, w_2, \ldots, w_l \rangle$ provided to the agent at the beginning of its navigation, where w_i denotes the i-th word in the sequence. The environment is defined as set of viewpoints $\{p_j\}$ denoting all the navigable locations. At time step t, the agent at viewpoint p_t receives a panoramic view $\boldsymbol{V}_t$ composed of n single view images $\langle \boldsymbol{v}_{t,1}, \boldsymbol{v}_{t,2}, \ldots, \boldsymbol{v}_{t,n} \rangle$. Using the given instruction $\boldsymbol{w}$ and the current observation $\boldsymbol{V}_t$, the agent needs to infer an action a_t which triggers a transition signal from p_t to p_{t+1}. The episode ends when the agent output a $STOP$ action or the maximum number of steps allowed is reached.

3.1 Base Agent Model

We build our sub-instruction module based on the current state-of-the-art VLN agents, as shown in Figure 2. Those agents share a similar pipeline, a sequence-to-sequence architecture with textual and visual attentions. In this section, we refer to the Self-Monitoring Agent (Ma et al., 2019a) to present the flow of information in the network.

Visual and textual encoding. Before the start of navigation, the agent first encodes the given instruction, using an LSTM with a learned embedding as $\hat{\boldsymbol{w}}_j = \text{Embed}(w_j)$ and $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_l = \text{LSTM}(\hat{\boldsymbol{w}}_1, \hat{\boldsymbol{w}}_2, \ldots, \hat{\boldsymbol{w}}_l)$, where $\boldsymbol{u}_j$ is the hidden state of word w_j in the instruction. In the case of the panoramic view, the agent encodes the images using a ResNet-152 model (He et al., 2016) pre-trained on ImageNet (Russakovsky et al., 2015)

for each navigable direction. A 4-dimensional vector $[\sin\psi, \cos\psi, \sin\theta, \cos\theta]$ is concatenated with the image encoding to represent the direction of visual features, where ψ and θ are the heading and elevation angles, respectively.

Policy module and co-grounding. We define the agent's state at time t as a combination of the attended textual representation $\hat{\boldsymbol{u}}_t$, the attended visual representation $\hat{\boldsymbol{v}}_t$ and the previous selected action $\boldsymbol{a}_{t-1}$, encoded by an LSTM as

$$\boldsymbol{h}_t, \boldsymbol{m}_t = \text{LSTM}([\hat{\boldsymbol{v}}_t; \boldsymbol{a}_{t-1}], (\hat{\boldsymbol{u}}_t, \boldsymbol{m}_{t-1})). \quad (1)$$

We refer to $\boldsymbol{h}$ and $\boldsymbol{m}$ as the agent's state and memory, respectively.

The attended textual representation is obtained by performing soft-attention over the language features $\boldsymbol{U}=\langle \boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_l \rangle$ with the agent's state at the previous time step. The attention weights over all the words are calculated as $z^{\text{text}}_{t,j}=(\boldsymbol{W}_u \boldsymbol{h}_{t-1})^T \boldsymbol{u}_j$ and $\boldsymbol{\alpha}_t=$ Softmax$(\boldsymbol{z}^{\text{text}}_t)$, obtaining the attended textual representation by $\hat{\boldsymbol{u}}_t=\boldsymbol{\alpha}_t{}^T \boldsymbol{U}$. Similarly, we perform soft-attention over the single-view visual features $\boldsymbol{V_t}$ as $z^{\text{vis}}_{t,i}=(\boldsymbol{W}_v \boldsymbol{h}_{t-1})^T g(\boldsymbol{v}_{t,i})$ where $g(\cdot)$ is a multi-layer perceptron (MLP), and the attention weight $\boldsymbol{\beta}_t=$ Softmax$(\boldsymbol{z}^{\text{vis}}_t)$. The attended visual representation is $\hat{\boldsymbol{v}}_t=\boldsymbol{\beta}_t{}^T \boldsymbol{V_t}$. The previous selected action $\boldsymbol{a}_{t-1}$ is represented by the visual features at the previously selected action direction. Finally, the agent decides an action by finding the visual features at a navigable direction with the highest correspondence to the attended language features $\hat{\boldsymbol{u}}$ and the agent's current state $\boldsymbol{h}_t$. The probability at each navigable direction is computed as:

$$o_{t,i} = (\boldsymbol{W}_a[\boldsymbol{h}_t, \hat{\boldsymbol{u}}_t])^T g(\boldsymbol{v}_{t,i}) \quad (2)$$

and

$$\boldsymbol{p}_t = \text{Softmax}(\boldsymbol{o}_t) \quad (3)$$

where $g(\cdot)$ is the same MLP as in visual attention for feature projection. Then, the agent moves in a panoramic action space (Fried et al., 2018), so that it jumps directly to an adjacent viewpoint in the selected direction.

All baseline agents in our experiments are variants of this pipeline. For instance, the Speaker-Follower agent (Fried et al., 2018) encodes the agent's state with only the previous action and the attended visual features. In the case of the Back-Translation agent (Tan et al., 2019), it attends the language features by the agent's current state.

3.2 Chunking

To encourage the learning of vision and language correspondences, we provide short and easier-to-learn sub-instructions to the agent at each time step. Formally, for each instruction $\boldsymbol{w}$, there exists a set of sub-instructions $\boldsymbol{X}=\langle \boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_L \rangle$, where $\boldsymbol{x}_i=\langle w_j \rangle$ and L is the total number of sub-instructions. The sub-instructions are ordered, mutually exclusive and cover the entire $\boldsymbol{w}$.

We propose a chunking function to break the original instruction into several sub-instructions, where each sub-instruction is an independent navigation task and usually requires the agent to perform one or two actions to complete. To achieve this automatically, we design chunking rules based on the grammatical relations between words in the instruction, where the relations are produced by the Stanford NLP Parser (Qi et al., 2018), a pre-trained natural language analysis tool. First, we pass the entire instruction into the StanfordNLP Parser for extracting the *dependency* and the *governor* of each word, denoted as $\eta(w_j)$ and $\rho(w_j)$, respectively. Then, using the two attributes, we formulate a heuristic as shown in Algorithm 1.

Algorithm 1 Chunking Function

Initialize empty lists l_{conj}, l_x, l_η, $l_{\boldsymbol{X}}$. Count $k = 0$.
# Find index of the word that satisfies condition (2)
for w_j in $\boldsymbol{w}$ **do**
  if $\eta(w_j)$ is `conj` && $\rho(w_j)$ is 1 **then**
    Save word index j into l_{conj}
  end if
end for
for w_j in $\boldsymbol{w}$ **do**
  # Condition (1)
  if $\eta(w_j)$ is `root` && (`root` in l_η or `parataxis` in l_η) **then**
    $l_{\boldsymbol{X}} \leftarrow$ **Check**(l_x)
    # Condition (2)
  else if $k \leq \text{len}(l_{conj}) - 1$ && $\rho(w_j)$ is $l_{conj}[k]$ **then**
    $l_{\boldsymbol{X}} \leftarrow$ **Check**(l_x), $k = k + 1$
    # Condition (3)
  else if $\eta(w_j)$ is `parataxis` && (`root` in l_η or `parataxis` in l_η) **then**
    $l_{\boldsymbol{X}} \leftarrow$ **Check**(l_x)
    # Save the word into temporary chunk
  else if $\eta(w_j)$ is not `punct` **then**
    Save w_j into l_x, save $\eta(w_j)$ into l_η
  end if
end for

The chunking function considers words in the instruction that meet one of the following three conditions as the beginning of a new sub-instruction: (1) its dependency is `root` and all the words before belong to the previous chunk, (2) its dependency is `conj` and its governor is the previous `root`, (3) its dependency is `parataxis` and all the words

before belong to the previous chunk. If any one of the three conditions is met, a **Check**$(\cdot)$ function will be performed on the temporary chunk to decide whether to save the temporary chunk into the final sub-instruction list $l_{\boldsymbol{X}}$. Here, the **Check**$(\cdot)$ function examines if the temporary chunk meets two conditions: (1) the chunk length should exceed the minimum length of two words, and (2) the temporary chunk should not only contains a single action-related phrase which is following the previous chunk or is leading the next chunk (e.g. "*go straight then ...*"), if it happens, then the temporary chunk should be appended to the previous chunk or added to the next chunk respectively.

We provide an illustrative example here. Our chunking function breaks the given instruction "*Enter through the glass door. Go up the wooden plank stairs on the right. Enter the doorway next to the bear head and wait there.*" into 1 "*Enter through the glass door*", 2 "*Go up the wooden plank stair on the right*", 3 "*Enter the doorway next to the bear head*" and 4 "*And wait there*", as shown in Figure 1. In the third and the fourth sub-instructions, the words "Enter" and "Wait" satisfy the conditions (1) and (2), respectively. Notice that the *governor* of conjunction word "And" is "Wait", so it has been assigned to the fourth sub-instruction.

3.3 Sub-Instruction Module

To encourage the agent to learn the correspondence between visual and language features in a sub-instruction, we modify the base agents to include a sub-instruction module, which enables the agent to focus on a particular sub-instruction at each time step, as shown in Figure 2. It contains two main components: the sub-instruction attention and the sub-instruction shifting module.

Sub-instruction attention. The module attends the words inside the selected sub-instruction $\boldsymbol{x}_i$ through a soft-attention mechanism. Formally, at each time step, we calculate the distribution of weights over each word in $\boldsymbol{x}_i$ as:

$$z_{t,j}^{\text{text}} = (\boldsymbol{W}_u \boldsymbol{h}_{t-1})^T \boldsymbol{x}_{i,j}, \qquad (4)$$
$$\boldsymbol{\alpha}_t = \text{Softmax}(\boldsymbol{z}_t^{\text{text}})$$

where $\boldsymbol{h}_{t-1}$ is the previous state of the agent and $\boldsymbol{W}_u$ is the learned weights. The grounded representation of the sub-instruction is hence $\hat{\boldsymbol{x}}_i = {\boldsymbol{\alpha}_t}^T \boldsymbol{x}_i$.

With the sub-instruction attention, the agent is forced to attend the most relevant part of the instruction and prevent the agent from "getting distracted" by the other part of the instruction that has been completed or to be completed in the further steps.

Sub-instruction shifting. At each time step, the agent needs to decide whether the current sub-instruction will be completed by the next action or not. We enable this functionality by designing a shifting module that estimates the probability of proceeding to the next sub-instruction.

The module uses a recurrent neural architecture to encode a representation that reflect the vision and language co-grounded features:

$$\boldsymbol{h}_t^c = \sigma(\boldsymbol{W}_{c1}[\boldsymbol{W}_{c0}(\boldsymbol{h}_t), \boldsymbol{v}_t^a, \hat{\boldsymbol{x}}_i]) \odot \tanh(\boldsymbol{m}_t) \quad (5)$$

where $\boldsymbol{h}_t$ and $\boldsymbol{m}_t$ is the agent's current state and memory, $\boldsymbol{v}_t^a$ is the visual feature at the selected action direction, σ represents a sigmoid function, $\boldsymbol{W}_{c1}$ and $\boldsymbol{W}_{c0}$ are the learned weights and $\odot$ denotes the Hadamard product.

The module then computes the shifting probability from $\boldsymbol{h}_t^c$ and a one-hot encoding $\boldsymbol{e}_t$ of the number of sub-instructions left to be completed, as:

$$p_t^s = \sigma(\boldsymbol{W}_{c2}[\boldsymbol{W}_{c3}(\boldsymbol{e}_t), \boldsymbol{h}_t^c]) \qquad (6)$$

where $\boldsymbol{W}_{c2}$ and $\boldsymbol{W}_{c3}$ are the learned parameters. Here, $\boldsymbol{e}_t$ introduces a learnable prior on when to shift before viewing the scene. This prior is then modified by taking into account the visual evidence, which is essential for efficient navigation. If the shifting probability exceed a certain threshold, a shift signal $s_t{=}1$ ($s_t{\in}\{0, 1\}$) of reading the next sub-instruction will be produced. We only enable the module to do a single step uni-directional shifting, which agrees with the fact that instructions and trajectory in the R2R dataset are monotonically aligned.

3.4 Training

In the training stage, for each instruction $\boldsymbol{w}$, there exists a corresponding ground-truth path $\boldsymbol{p}_g = \langle p_{g(1)}, p_{g(2)}, ..., p_{g(M)} \rangle$. In the case of sub-instructions, we partition the path into sub-paths, one for each sub-instruction.

The binary cross-entropy loss compares the estimated shifting probabilities p_t^s to the target shifting signals y_t^s, where the target is either 1 or 0 depending on the distance between the agent's current position and the ending viewpoint of the current

sub-path. In summary, the agent's parameters are learned to optimized

$$\mathcal{L} = -\sum_t y_t^a \log p_t^a - \sum_t y_t^s \log p_t^s + (1 - y_t^s) \log(1 - p_t^s) \quad (7)$$

where p_t^a is the predicted action, y_t^a and y_t^s are the ground-truth action and shifting signal respectively at time step t.

During training, we apply student-forcing supervision to the action to encourage exploration, but use teacher-forcing for the sub-instruction shifting (Williams and Zipser, 1989; Anderson et al., 2018b). In early stages of training, the ground-truth shifting signal will have a large number of zeros since the agent has a high probability of deviating from the desired path. We prevent the sub-instruction shifting module from converging to an undesirable local minimum by forcing the shifting loss to consider an equal number of randomly selected shift and do-not-shift samples in each time step.

4 The FGR2R Dataset

To acquire the matching between vision and language sub-sequences, we introduce a Fine-Grained Room-to-Room (FGR2R) dataset which enriches the benchmark Room-to-Room dataset by dividing the instructions into sub-instructions and pairing each of those with their corresponding viewpoints in the path.

Dataset collection. We first apply the chunking function introduced in Section 3.2 to generate the sub-instructions automatically from the original R2R data. We demonstrate the quality of the generated sub-instructions by comparing the output sub-instructions against a manually annotated subset of 300 samples, obtaining a smoothed BLEU-4 score of 0.84. Then, we add annotations of sub-path corresponding to each sub-instruction using the Amazon Mechanical Turk (AMT)[2]. We refer the readers to Appendix A.1 for more information about the data collection interface and the qualification process of the annotators that we designed to ensure the quality of the collected data.

Dataset statistics. The original R2R possesses 21,567 navigation instructions and 7,189 paths in 91 real-world environments, where 3 or 4 different natural language instructions describe each path. The R2R data has been split for learning proposes, with 4,675 paths for training and 340 paths for seen validation in 61 scenes, 783 paths in 11 scenes for unseen validation and the remaining 1,391 paths in 18 scenes for testing[3]. Based on the original R2R data, FGR2R divides the instructions for the training and validation set in an average of 3.6 sub-instructions. Each sub-instruction has 7.2 words on average. Sub-instructions are paired on average 2.4 viewpoints, and with a minimum and maximum of 1 and 7 viewpoints, respectively. We refer the readers to Appendix A.1 for more dataset statistics.

5 Experiments

5.1 Experiment Setup

We experiment with four state-of-the-art VLN agents with and without our sub-instruction module and compare their performance on the original R2R validation unseen split.

The agents are chosen to include the most common network architectures, training strategies and inference methods among the previous VLN agents. They include the Sequence-to-Sequence (Seq2Seq) (Anderson et al., 2018b) model which does not apply panoramic action space, two visual-textual co-grounding models, the Speaker-Follower (Fried et al., 2018) and the Self-Monitoring agent (Ma et al., 2019a), as well as the Back-Translation model (Tan et al., 2019) which applies reinforcement learning. For all agents, we implement our sub-instruction module in their network based on their officially released code. For the self-monitoring agent, we remove the progress monitor since it requires the attention weight over the entire instruction for estimating the navigation progress.

Implementation details. To obtain the word representations in each sub-instruction, the entire instruction is first passed to a unidirectional LSTM, then we implement chunking on the language hidden states to obtain the word representations of the selected sub-instructions. The ground-truth shifting signal at each time-step is dependent on the distance between the agent's current position and the end viewpoint of the selected sub-instruction. If the distance is smaller than or equal to 0.5 meters, the ground-truth shift signal s_t will be 1, and 0 otherwise. For the Back-Translation model (Tan et al., 2019), we only apply chunk shifting loss to the

[2]Amazon Mechanical Turk: https://www.mturk.com/

[3]More information about R2R can be found in the Matterport3D dataset(Chang et al., 2017) and the R2R dataset (Anderson et al., 2018b)

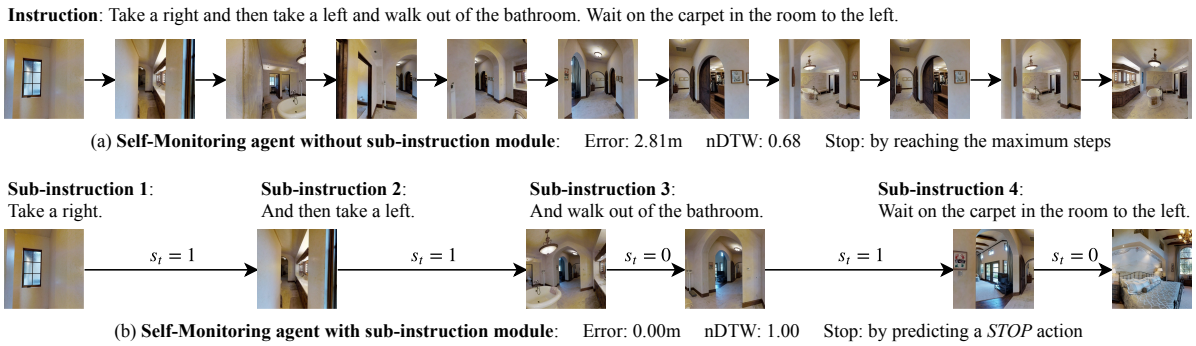


Figure 3: Qualitative comparison of a successful case without and with sub-instruction module. Without sub-instruction module, the agent fails to follow the instruction and stops next to the target by chance. With sub-instruction module, the agent navigates on the described path and eventually stops right at the target location. For panoramic visualization and more examples please refer to the supplementary material.

		R2R Validation Unseen					
#	Model	PL ↓	NE ↓	OSR ↑	SR ↑	SPL ↑	nDTW ↑
1	Seq2Seq (Anderson et al., 2018b)	**8.34** (8.71)	**7.85** (7.92)	29.2 (**29.5**)	**22.9** (21.8)	**0.20** (0.18)	**0.58** (0.57)
2	Speaker-Follower (Fried et al., 2018)	**13.57** (16.66)	**6.66** (7.12)	**44.8** (41.1)	**34.7** (29.8)	**0.28** (0.22)	**0.59** (0.54)
3	Self-Monitoring (Ma et al., 2019a)	**13.95** (15.02)	**6.16** (6.29)	**53.7** (53.0)	**42.4** (40.7)	**0.32** (0.30)	**0.61** (0.58)
4	Back-Translation (Tan et al., 2019)	9.81 (**9.62**)	5.67 (**5.61**)	54.8 (**54.9**)	**46.7** (46.6)	**0.43** (**0.43**)	0.69 (**0.70**)

Table 1: Comparison on the validation unseen split with and without the sub-instruction module. Values not in brackets are with sub-instructions, values in brackets are without sub-instructions.

teacher-forcing imitation learning branch, so that the agent navigates on the ground-truth path and learns the chunk-shifting with less noise. We train all agents on a single NVIDIA Tesla K80 GPU, using the same hyperparameters as the baselines.

Evaluation metrics. We follow the standard metrics that previous work employed for evaluating the agent's performance on the R2R dataset (Anderson et al., 2018b), which include Path Length (PL) of the agent's trajectory, average Navigation Error (NE) for the distance between agent's final position and the target, Oracle Success Rate (OSR) for the ratio of agents which the shortest distance between the target and the trajectory is within $3m$, Success Rate (SR) for the ratio of agents which the distance between agent's final position and the target is within $3m$, and Success Rate Weighted by Path Length (SPL). Furthermore, we also consider the normalized Dynamic Time Warping (nDTW) score (Magalhaes et al., 2019), which is a metric that measure the overall performance of the agent with a focus on the similarity between the ground-truth and the actual trajectories.

6 Results and Analysis

We compare the performance of the four agents on the R2R unseen validation set. We also present the traceability of the navigation process resulting from our FGR2R data.

6.1 Comparisons

Quantitative results. Table 1 shows the results of the four agents in unseen environments. The performance of the imitation learning agents (Row 1–3) with our sub-instruction attention module outperforms the base agents. In terms of the success rate, the Seq2Seq, Speaker-Follower and the Self-Monitoring agents achieve an absolute increase of 1.1%, 4.9% and 1.7% respectively. The improvement is consistent in most of the other metrics, e.g. for the Self-Monitoring agent, its SPL improves from 0.30 to 0.32 and its nDTW score grows from 0.58 to 0.61. The overall improvement on Path Length and nDTW score for the first three agents indicates that using sub-instructions improves the agent's ability to navigate on the described path. As for the Back-Translation agent (Row 4), the performance with sub-instruction attention is very similar to the baseline, one possible reason could be that the introduction of sub-instruction shifting perturbs

#	Model with sub-instructions	SR	TP	TN	FP	FN	Accuracy	Precision	Recall	F1-Score
1	Seq2Seq (Anderson et al., 2018b)	22.9	608	36344	1602	4796	0.852	0.275	0.113	0.160
2	Speaker-Follower (Fried et al., 2018)	34.7	963	9966	452	4878	0.672	0.681	0.165	0.265
3	Self-Monitoring (Ma et al., 2019a)	42.4	1130	10619	363	4686	0.699	0.757	0.194	0.309
4	Back-Translation (Tan et al., 2019)	46.7	1256	8086	303	4765	0.648	0.806	0.209	0.331

Table 2: Statistics of the shifting signal on the unseeen validation set.

the learning of action during for the reinforcement learning scheme which the agent could deviate far from the ground-truth path.

Learning when the agent needs to read a new sub-instruction is a difficult task, the same viewpoint in a specific environment can be considered as a shifting point or not depending on the sub-instruction that the agent follows. In Table 2, we show the confusion matrix of the shifting signals and we compute accuracy, precision, recall and F1-score to evaluate the performance of our proposed shifting module. Results show that all the agents have huge room for improvement for shifting, since the best F1-Score obtained is only 0.331. But we can see from the four agents that, as the success rate increases, the precision, recall and F1-score also improve. We propose to consider these results to be useful baselines for future methods that apply sub-instructions. Notice that agents visit a different number of viewpoints due to the maximum number of steps allowed, the use of panoramic action space and the ability to stop. In the case of Seq2Seq model, since the agent is not using a panoramic view, it performs many actions to change the camera orientation.

Qualitative performance. We illustrate a qualitative example in Figure 3 to show how the sub-instruction module works in the agent. In the example, both the baseline model and the model with the sub-instruction module completes the task successfully. However, unlike the baseline model which fails to follow the instruction and stops within 3 meters of the target by chance, our model correctly identifies the completeness of each sub-instruction, guides the agent to walk on the described path and eventually stops right at the target position. We refer the readers to Supplementary Materials for visualization of more trajectories.

6.2 Traceability

With the FGR2R data, we reveal the navigation process of the agent working on specific sub-instructions. For each sub-instruction, we measure the similarity between the ground-truth path and the actual trajectory using nDTW as well as the distance between the end viewpoint of the sub-instruction and the predicted shift viewpoint. As a result, we can estimate the performance of the agent in each sub-task.

rank	$\overline{d}$	$\overline{nDTW}$	f	$\overline{s}$	Representative sub-instruction
1	2.22	0.72	7	2.8	head down the stair
2	2.52	0.57	5	4.6	wait near the first open door
3	2.58	0.73	8	2.5	go into the bedroom
4	2.66	0.73	21	2.6	exit the bedroom
5	2.77	0.65	10	2.1	turn right at the entry
⋮	⋮	⋮	⋮	⋮	⋮
96	6.33	0.55	35	3.8	stop behind the table at the far end
97	6.56	0.43	8	2.0	walk past the sink, fridge, oven
98	6.86	0.46	11	3.2	go through the wooden archway
99	6.88	0.51	20	3.0	walk along the grass until you reach ...
100	7.36	0.52	38	3.4	walk into the room which have a ...

Table 3: Performance on different sub-instruction clusters in validation unseen split. $\overline{d}$, f and $\overline{s}$ denote the mean distance, the frequency and the mean number of viewpoints of a cluster.

We cluster the sub-instructions into 100 clusters using complete-linkage hierarchical agglomerative clustering algorithm. Instead of using a standard metric of distance such as the Euclidean distance, we compute a similarity matrix of sub-instructions using the BLEU-4 metric. We experiment with the Self-Monitoring agent on validation unseen split and present a summary of the top five and the bottom five clusters ranked by the mean distance, as shown in Table 3.

We can see from the table that the clusters which the agent performs better consist of simple and direct sub-instructions which refer to a single action, such as "*head down the stair*" and "*exit the bedroom*". On the other hand, with sub-instructions that refer to specific objects such as "*walk past the sink, fridge, oven*" or express an action which is conditioned on the completion of another action, such as "*walk along the grass until you reach ...*", the agent deviates far from the described path. Moreover, the ranking does not show a strong correlation with the frequency or the number of viewpoints of each sub-instruction. These results suggest that agent is incapable of understanding complex natural language instructions or ground to specific objects with a high accuracy.

7 Conclusion

In this paper we introduce a novel sub-instruction module and the Fine-Grained R2R Dataset to encourage the learning of correspondences between vision and language. The sub-instruction module enables the agent to attend to one particular sub-instruction at each time-step and decides whether the agent needs to proceed to the next sub-instruction. Our experiments show that by implementing the sub-instruction module in state-of-the-art agents, most of the agents are able to follow the given instruction more closely and achieve better performance. We also show that, with the sub-instruction annotations, the entire navigation trajectory is trackable. We believe that the idea of sub-instruction module and a sub-instruction annotated dataset can benefit future studies in the VLN task as well as other vision-and-language problems.

References

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018a. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6077–6086.

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. 2018b. 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 Niebner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. 2017. Matterport3d: Learning from rgb-d data in indoor environments. In *2017 International Conference on 3D Vision (3DV)*, pages 667–676. IEEE.

Howard Chen, Alane Suhr, Dipendra Misra, Noah Snavely, and Yoav Artzi. 2019. Touchdown: Natural language navigation and spatial reasoning in visual street environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12538–12547.

Marcella Cornia, Lorenzo Baraldi, and Rita Cucchiara. 2019. Show, control and tell: A framework for generating controllable and grounded captions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8307–8316.

Martin Engilberge, Louis Chevallier, Patrick Pérez, and Matthieu Cord. 2018. Finding beans in burgers: Deep semantic-visual embedding with localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3984–3993.

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. In *Advances in Neural Information Processing Systems*, pages 3314–3325.

Jiyang Gao, Chen Sun, Zhenheng Yang, and Ram Nevatia. 2017. Tall: Temporal activity localization via language query. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5267–5275.

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.

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.

Haoshuo Huang, Vihan Jain, Harsh Mehta, Alexander Ku, Gabriel Magalhaes, Jason Baldridge, and Eugene Ie. 2019. Transferable representation learning in vision-and-language navigation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7404–7413.

Drew Hudson and Christopher D Manning. 2019. Learning by abstraction: The neural state machine. In *Advances in Neural Information Processing Systems*, pages 5901–5914.

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.

Liyiming Ke, Xiujun Li, Yonatan Bisk, Ari Holtzman, Zhe Gan, Jingjing Liu, Jianfeng Gao, Yejin Choi, and Siddhartha Srinivasa. 2019. Tactical rewind: Self-correction via backtracking in vision-and-language navigation. In *Proceedings of the*

IEEE Conference on Computer Vision and Pattern Recognition, pages 6741–6749.

Federico Landi, Lorenzo Baraldi, Massimiliano Corsini, and Rita Cucchiara. 2019. Embodied vision-and-language navigation with dynamic convolutional filters. *Proceedings of the British Machine Vision Conference*.

Chih-Yao Ma, Asim Kadav, Iain Melvin, Zsolt Kira, Ghassan AlRegib, and Hans Peter Graf. 2018. Attend and interact: Higher-order object interactions for video understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6790–6800.

Chih-Yao Ma, Yannis Kalantidis, Ghassan AlRegib, Peter Vajda, Marcus Rohrbach, and Zsolt Kira. 2020. Learning to generate grounded visual captions without localization supervision. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Chih-Yao Ma, Jiasen Lu, Zuxuan Wu, Ghassan AlRegib, Zsolt Kira, Richard Socher, and Caiming Xiong. 2019a. Self-monitoring navigation agent via auxiliary progress estimation. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

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 Conference on Computer Vision and Pattern Recognition*, pages 6732–6740.

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

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937.

Peng Qi, Timothy Dozat, Yuhao Zhang, and Christopher D Manning. 2018. Universal dependency parsing from scratch. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 160–170.

Cristian Rodriguez, Edison Marrese-Taylor, Fatemeh Sadat Saleh, Hongdong Li, and Stephen Gould. 2020. Proposal-free temporal moment localization of a natural-language query in video using guided attention. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 2464–2473.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

Idan Schwartz, Alexander Schwing, and Tamir Hazan. 2017. High-order attention models for visual question answering. In *Advances in Neural Information Processing Systems*, pages 3664–3674.

Hao Tan, Licheng Yu, and Mohit Bansal. 2019. Learning to navigate unseen environments: Back translation with environmental dropout. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2610–2621.

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 Conference on Computer Vision and Pattern Recognition*, pages 6629–6638.

Xin Wang, Wenhan Xiong, Hongmin Wang, and William Yang Wang. 2018. Look before you leap: Bridging model-free and model-based reinforcement learning for planned-ahead vision-and-language navigation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 37–53.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057.

Licheng Yu, Zhe Lin, Xiaohui Shen, Jimei Yang, Xin Lu, Mohit Bansal, and Tamara L Berg. 2018. Mattnet: Modular attention network for referring expression comprehension. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1307–1315.

Fengda Zhu, Yi Zhu, Xiaojun Chang, and Xiaodan Liang. 2020a. 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.

Wang Zhu, Hexiang Hu, Jiacheng Chen, Zhiwei Deng, Vihan Jain, Eugene Ie, and Fei Sha. 2020b. Baby-Walk: Going farther in vision-and-language navigation by taking baby steps. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2539–2556. Association for Computational Linguistics.

A Appendices

A.1 FGR2R Dataset

Data collection. We build a web interface to collect FGR2R data using Amazon Mechanical Turk (AMT), as shown in Figure 5. In the interactive window, each viewpoint on the ground-truth path is highlighted with a large cylinder and an index of the viewpoint. Besides each sub-instruction, there is a drop-down list for assigning the start and end viewpoints of the corresponding sub-path. The annotators can click in the interactive window to freely move on the ground-truth path and freely rotate the camera to observe its surroundings. Before the start of labelling, we first ask the annotators to watch the automatic trajectory run-through to get familiar with the environment. Then, we ask them to partition the ground-truth path and assign a sub-instruction to those partitions. Once the labelling is completed, a function will automatically check if the annotation disobeys any rules (e.g., the start viewpoint of a sub-path should be the same as the end viewpoint of the previous sub-path) before approval for submission.

Annotator qualification. To ensure the quality of the annotation returned by the annotators, we annotated a subset of 300 samples as ground-truths and we exam each annotator with 15 ground-truth samples before approval for labelling. In total, there are 126 participants. We reject workers with a low agreement to the ground-truth. The qualification process leaves us 58 qualified annotators to complete the annotation task.

Dataset statistics. Apart from the FGR2R statistics mentioned in the paper, we present the distribution of sub-instructions in an instruction and the distribution of viewpoints for a sub-instruction in Figure 4. As we can see, most of the instructions are broken down into more than one sub-instruction and the frequency of more than seven sub-instructions is very low. Also, notice that about 15% of the sub-instructions are paired with only one viewpoint, as a result of the sub-instructions that only refer to camera rotation such as "*rotate slightly to the left*" or stopping command such as "*wait by the sink*".

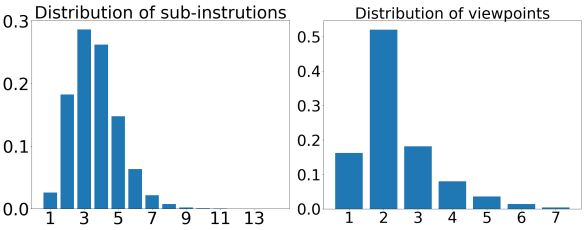


Figure 4: Distribution of sub-instructions in an instruction and distribution of viewpoints for a sub-instruction in the FGR2R dataset.

Training with FGR2R. During training, consider that more coherent motion could be beneficial for the agent to learn the textual-visual correspondence. We combine the sub-instructions which are only paired with one viewpoint to the next sub-instruction (and combine with the previous sub-instruction if it is the last one). The sub-instructions in validation sets remain in their original format so that the ground-truth trajectories are kept unknown. In this work, we only enable the sub-instruction module with a single step uni-directional shifting, which agrees with the observation that instructions and trajectory in the R2R dataset are monotonically aligned. However, different rules could be designed. For example, one can allow the agent to shift for more than one step or enable the agent to read the previous sub-instructions once it backtracks to the visited viewpoint. Our proposed FGR2R make all these research directions possible. We leave these ideas to future research.

A.2 Extension to Fine-Grained R4R

R2R to R4R The R4R dataset is created by concatenating two trajectories in R2R, which the first path ends within three meters from the start of the second path (Jain et al., 2019). We enrich the R4R data with sub-instructions annotations by joining two sequences of sub-instructions corresponding to the two trajectories. However, for some trajectories in R4R, there exist several additional viewpoints for connecting the two paths, which has no sub-instruction annotation. Therefore, we assign those additional viewpoints to the first sub-instruction of the second path.

Evaluation We further experimented the four agents on the R4R dataset, with and without sub-instruction modules. As shown in Table 4, the performance of the first three agents are very similar. For agents with sub-instruction modules, the SR of Seq2Seq and Speaker-Follower are slightly lower, whereas the SR of Self-Monitoring agent

#	Model	R4R Validation Unseen					
		PL	NE ↓	OSR ↑	SR ↑	SPL ↑	nDTW ↑
1	Seq2Seq (Anderson et al., 2018b)	9.40 (10.85)	9.35 (**9.20**)	32.8 (**35.5**)	21.2 (**22.3**)	**0.11** (**0.11**)	0.42 (**0.43**)
2	Speaker-Follower (Fried et al., 2018)	26.64 (25.68)	8.46 (**8.09**)	**42.1** (40.7)	26.4 (**27.4**)	0.12 (**0.13**)	**0.41** (**0.41**)
3	Self-Monitoring (Ma et al., 2019a)	28.01 (23.41)	**8.07** (8.46)	**46.2** (40.6)	**27.4** (25.8)	**0.10** (0.09)	**0.42** (0.41)
4	Back-Translation (Tan et al., 2019)	7.78 (39.66)	9.33 (**7.90**)	38.1 (**53.5**)	21.5 (**31.2**)	**0.17** (0.14)	**0.48** (0.39)

Table 4: Comparison on the R4R validation unseen split with and without the sub-instruction module. Values not in brackets are with sub-instructions, values in brackets are without sub-instructions.

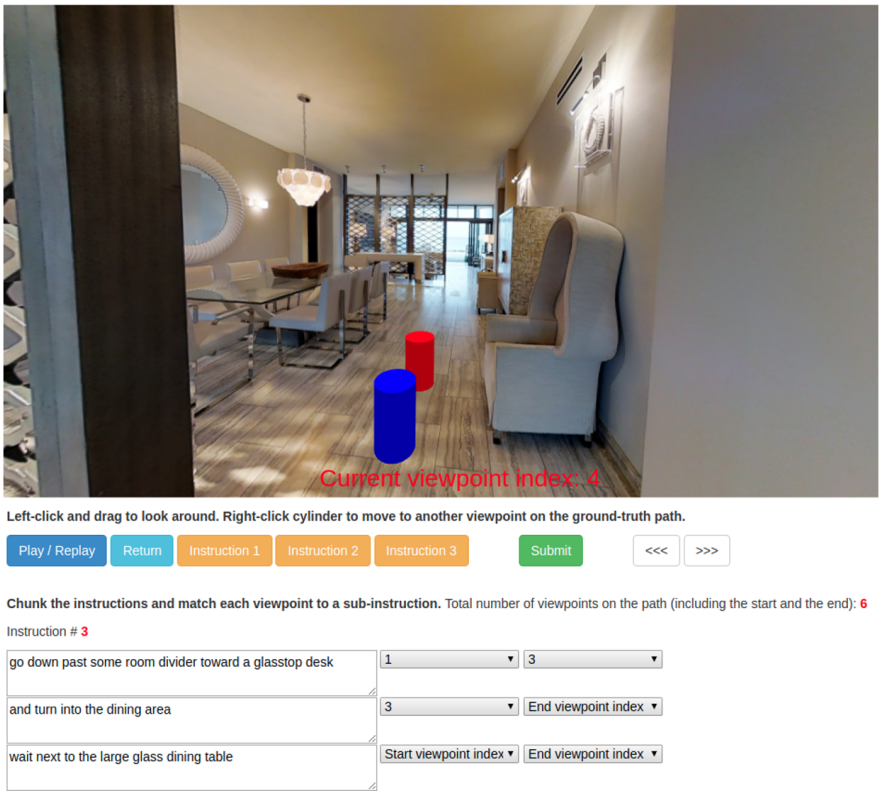


Figure 5: The web interface for FGR2R data collection. The displayed photo of the environment is an interactive window, cylinders are the viewpoints on the ground-truth path. "*Play / Replay*" shows an automatic run-through of the entire trajectory. "*Return*" brings the agent back to the first viewpoint. "*Instruction #*" switches among the three instructions that described the same path. "*Submit*" checks and submits the annotations.

is 1.6% higher. As for the Back-Translation, the agent experiences a large OSR and SR drop after applying sub-instructions, but the SPL and nDTW are increased by 3% and 9%. This result indicates that although the agent with sub-instruction modules has a lower chance to reach the target (stop within 3m), it follows the instruction much better.

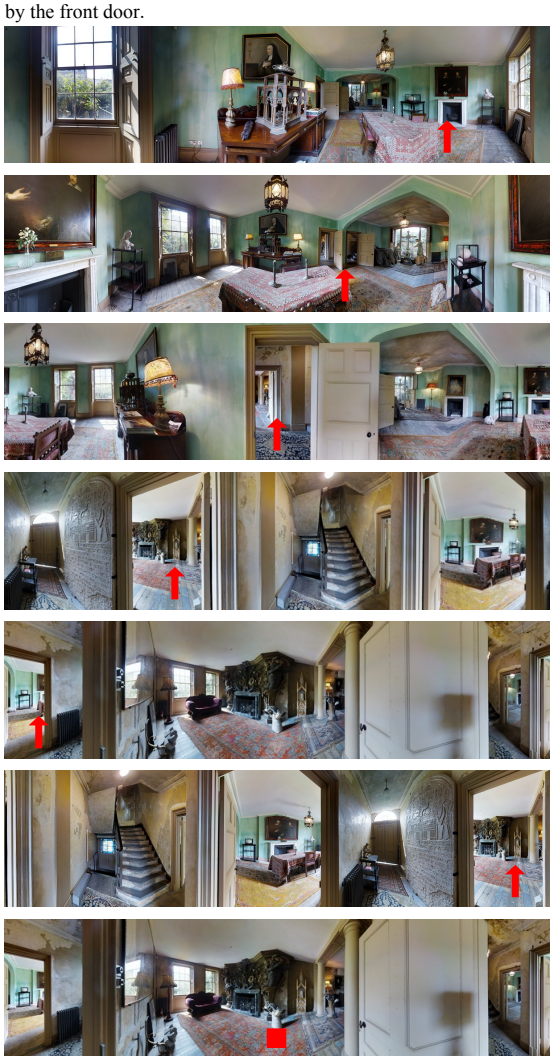
However, we argue that a large performance gain has not been obtained in R4R mainly for two reasons: (1) The additional viewpoints created for linking the two trajectories have no corresponding sub-instructions. Hence, agents trained to follow each sub-instructions strictly have no guidance for those steps. (2) The last sub-instruction of the first trajectory is very confusing to the agent, as it usually refers to the $STOP$ action, but the navigation does not end. This prevents the agent from learning a good stopping policy since the ground-truth action requires the agent to keep moving.

In conclusion, we believe that it is inappropriate to apply FGR2R data directly for FGR4R task. To obtain FGR4R data, our suggestion is to remove the final sub-instruction about the $STOP$ action from the first trajectory, and use a Speaker module (Fried et al., 2018) to generate a new sub-instruction for the additional viewpoints for linking the two trajectories. We will leave this idea for future work.

A.3 Visualization of Navigation

We visualize the navigation trajectories of the Self-Monitoring agent with and without our proposed sub-instruction module in the following pages.

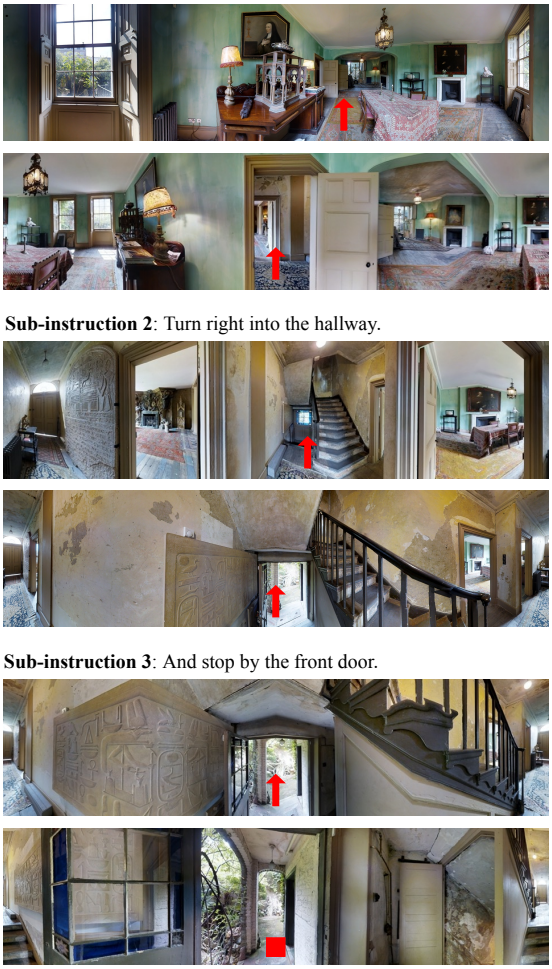
Instruction:
Go in the doorway on the left. Turn right into the hallway and stop by the front door.



Error: 5.11m nDTW: 0.61 Stop: by predicting a *STOP* action

(a) Self-Monitoring agent without sub-instruction attention

Sub-instruction 1: Go in the doorway on the left.



Sub-instruction 2: Turn right into the hallway.

Sub-instruction 3: And stop by the front door.

Error: 0.00m nDTW: 1.00 Stop: by predicting a *STOP* action

(b) Self-Monitoring agent with sub-instruction attention

Figure 6: A positive example of sub-instruction aware navigation. Without sub-instruction module, the agent wanders between rooms and decides to stop at a wrong location. With sub-instruction module, the agent successfully leaves the room, finds the way to the target and stops at the right location.

Instruction:
Take a right and then take a left and walk out of the bathroom. Wait on the carpet in the room to the left.

Error: 2.81m nDTW: 0.68 Stop: by reaching the maximum steps

(a) Self-Monitoring agent without sub-instruction module

Sub-instruction 1: Take a right.

Sub-instruction 2: And then take a left.

Sub-instruction 3: And walk out of the bathroom.

Sub-instruction 4: Wait on the carpet in the room to the left.

Error: 0.00m nDTW: 1.00 Stop: by predicting a *STOP* action

(b) Self-Monitoring agent with sub-instruction module

Figure 7: A positive example of sub-instruction aware navigation. Without sub-instruction module, the agent fails to follow the instruction and stops next to the target by chance. With sub-instruction module, the agent navigates on the described path and eventually stops right at the target location.

Instruction:
With the door leading outside behind you, walk forward and turn left to go down the corridor with the eye chart towards your right. Continue past the half bath on your left and the kitchen on your right, then turn left. Enter the bedroom ahead of your through the leftmost door on the opposite wall.

Error: 1.40m nDTW: 0.85 Stop: by reaching the maximum steps

(a) Self-Monitoring agent without sub-instruction attention

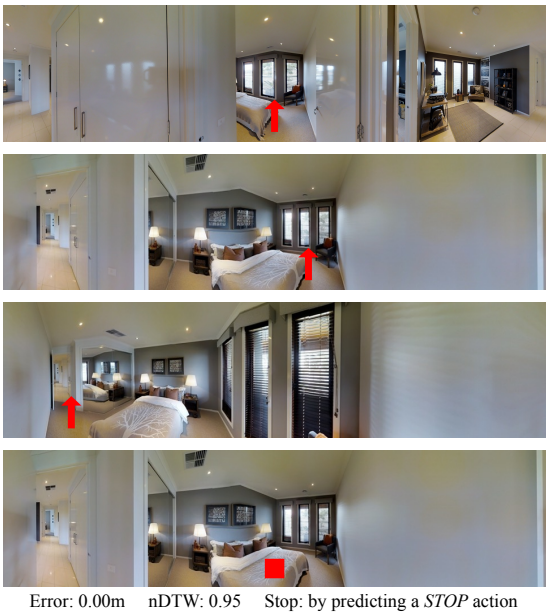
Sub-instruction 1: With the door lead outside behind you, walk forward.

Sub-instruction 2: And turn left to go down the corridor with the eye chart towards your right.

Sub-instruction 3: Continue past the half bath on your left and the kitchen on your right.

Sub-instruction 4: Then turn left.

Sub-instruction 5: Enter the bedroom ahead of you through the leftmost door on the opposite wall.

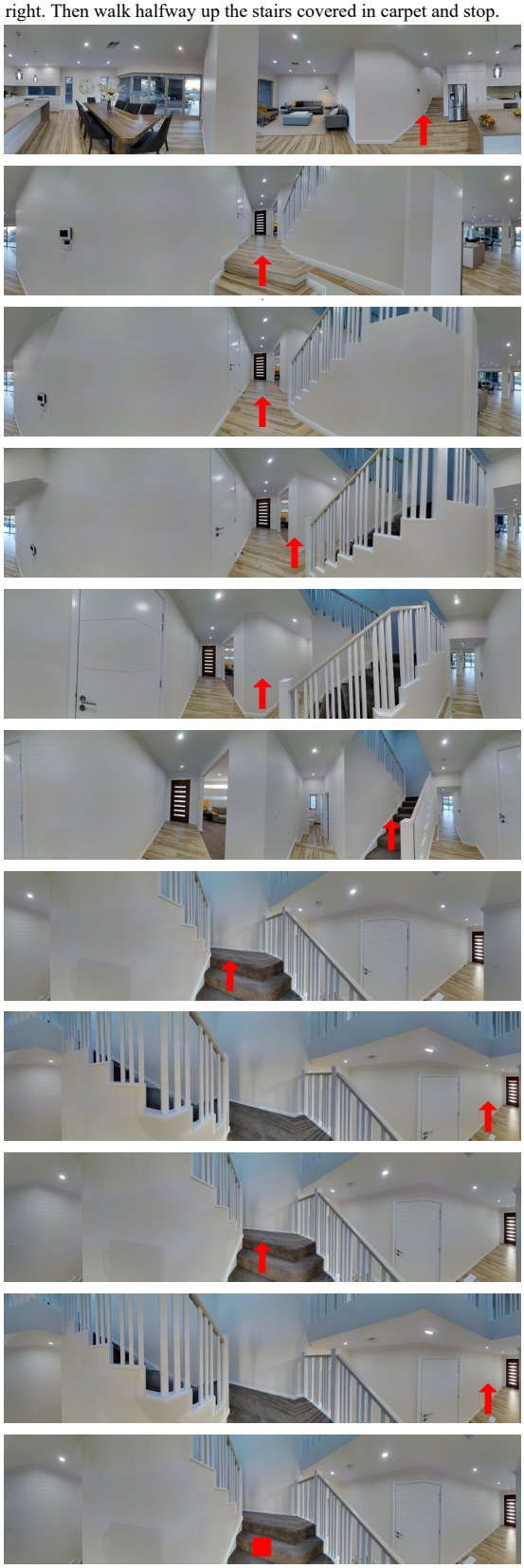


Error: 0.00m nDTW: 0.95 Stop: by predicting a *STOP* action

(b) Self-Monitoring agent with sub-instruction attention

Figure 8: A positive example of sub-instruction aware navigation. Without sub-instruction module, the agent loops around the target and doesn't know how to stop. With sub-instruction module, the agent falls into the same loop but quickly escapes from it and stops at the correct location.

Instruction:
Turn right and go up the wood stairs. At the top walk forward and turn right. Then walk halfway up the stairs covered in carpet and stop.



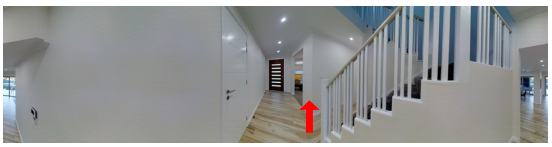
Error: 0.61m nDTW: 0.68 Stop: by reaching the maximum steps

(a) Self-Monitoring agent without sub-instruction attention

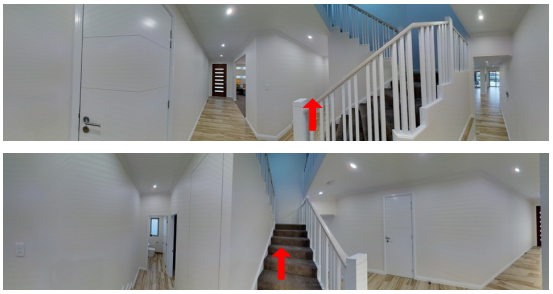
Sub-instruction 1: Turn right and go up the wood stair.

Sub-instruction 2: At the top walk forward.

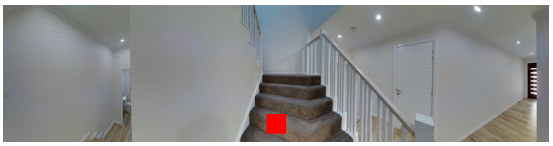
Sub-instruction 3: And turn right.



Sub-instruction 4: Then walk halfway up the stairs covered in carpet.



Sub-instruction 5: And stop.



Error: 0.00m nDTW: 1.00 Stop: by predicting a *STOP* action

(b) Self-Monitoring agent with sub-instruction attention

Figure 9: A positive example of sub-instruction aware navigation. Without sub-instruction module, the agent loops around the target and doesn't know how to stop. With sub-instruction module, the agent navigates on the described path and eventually stops right at the target location.

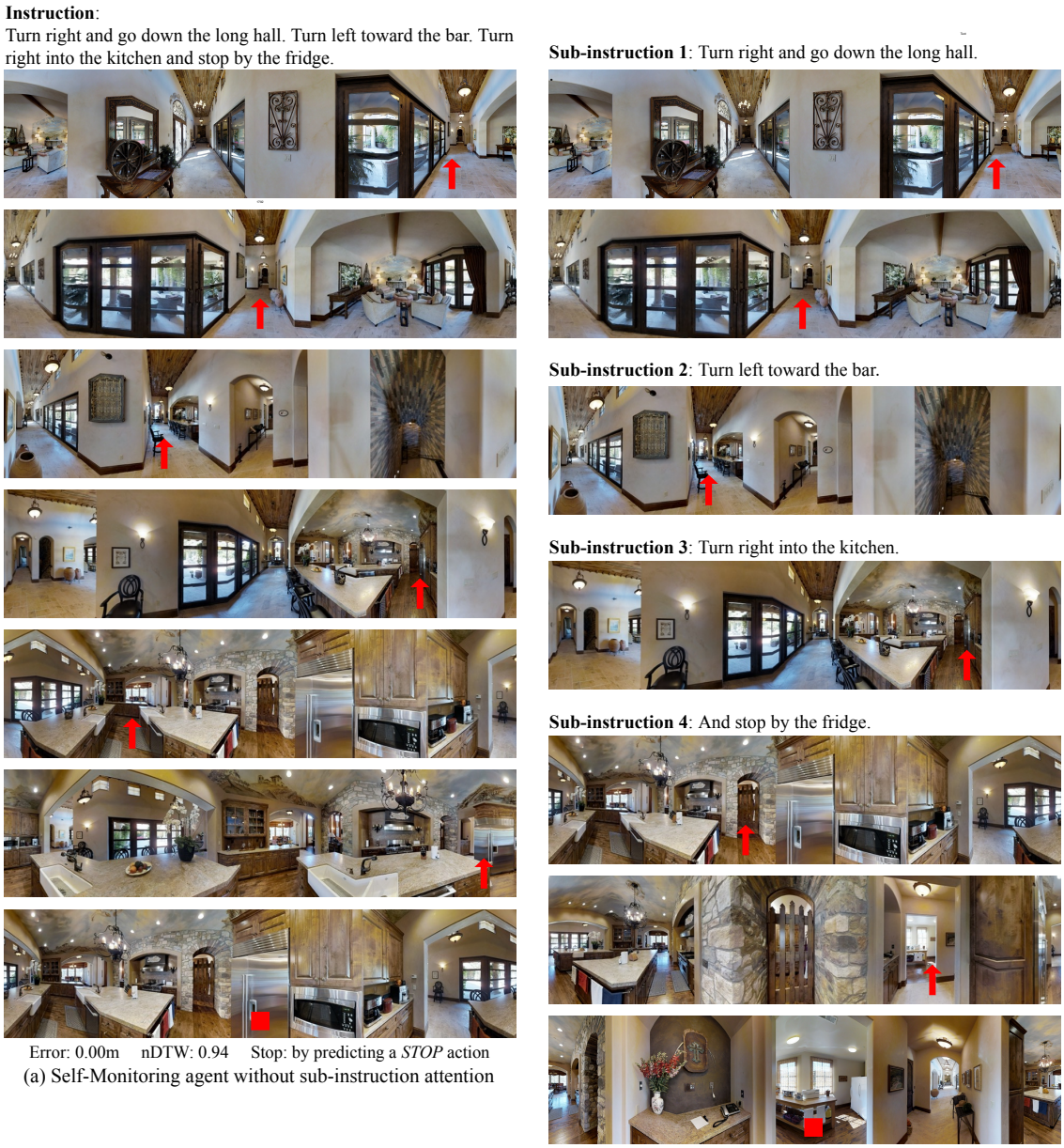


Figure 10: A negative example of sub-instruction aware navigation. Without sub-instruction module, the agent completes the navigation task without making any mistake. With sub-instruction module, although the agent performs sub-instruction shifting perfectly, it overlooks the target object and walks away from the target, eventually decides to stop at a wrong location.