Reinforced Structured State-Evolution for Vision-Language Navigation

Jinyu Chen[1,2], Chen Gao[1,2], Erli Meng[3], Qiong Zhang[3], Si Liu[1,2*]

[1]Institute of Artificial Intelligence, Beihang University

[2]Hangzhou Innovation Institute, Beihang University

[3]Xiaomi AI Lab, Xiaomi Inc

https://github.com/chenjinyubuaa/SEvol

Abstract

Vision-and-language Navigation (VLN) task requires an embodied agent to navigate to a remote location following a natural language instruction. Previous methods usually adopt a sequence model (e.g., Transformer and LSTM) as the navigator. In such a paradigm, the sequence model predicts action at each step through a maintained navigation state, which is generally represented as a one-dimensional vector. However, the crucial navigation clues (i.e., object-level environment layout) for embodied navigation task is discarded since the maintained vector is essentially unstructured. In this paper, we propose a novel Structured state-Evolution (SEvol) model to effectively maintain the environment layout clues for VLN. Specifically, we utilise the graph-based feature to represent the navigation state instead of the vector-based state. Accordingly, we devise a Reinforced Layout clues Miner (RLM) to mine and detect the most crucial layout graph for long-term navigation via a customised reinforcement learning strategy. Moreover, the Structured Evolving Module (SEM) is proposed to maintain the structured graph-based state during navigation, where the state is gradually evolved to learn the object-level spatial-temporal relationship. The experiments on the R2R and R4R datasets show that the proposed SEvol model improves VLN models' performance by large margins, e.g., $+3\%$ absolute SPL accuracy for NvEM and $+8\%$ for EnvDrop on the R2R test set.

1. Introduction

In recent years, Embodied-AI (E-AI) that requires embodied agents to complete tasks has arrested extensive interests of both computer version and natural language processing community. Numerous datasets [3, 31] have been constructed to simulate realistic environments to support various embodied tasks such as navigation [31, 42], interactive learning [6, 33] and multi-agent cooperation [25], *etc*.

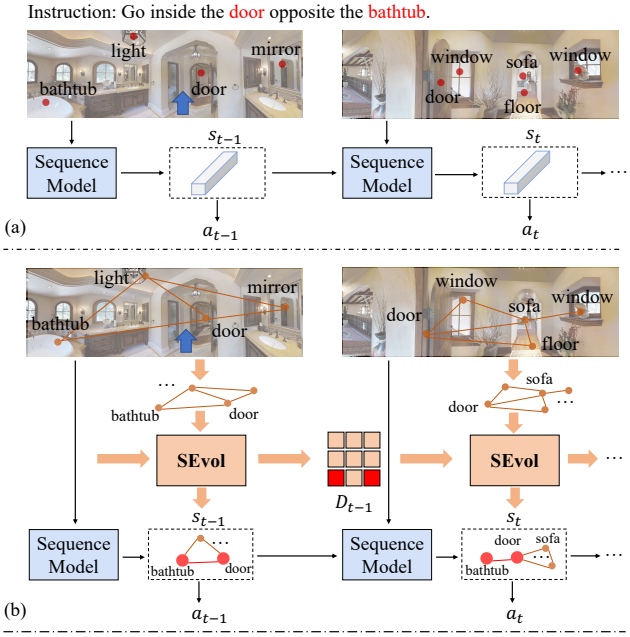

Figure 1. At each step t, (a) previous methods predict action a_t based on a *vector-based* navigation state s_t while the object-level layout memory is discarded; (b) we propose SEvol to maintain a *graph-based* navigation state s_t, which can effectively record the layout memory via the structured state-evolution.

One of the most attractive application scenarios of E-AI is the Vision-and-Language Navigation (VLN) task [2], where the goal is for an embodied agent in a 3D environment to navigate to the specific location following the natural language instruction. As shown in Figure 1(a), previous methods [8, 21, 22, 34] usually adopt sequence model (*e.g.*, Transformer and LSTM) to model the sequential decision process. At each step t, the action is predicted according to the navigation state s_t, which maintains the historical and current environment information. Commonly,

*Corresponding author: *Si Liu*.

the navigation state is maintained in the form of *unstructured one-dimensional vector*. The environment clues at each step, *i.e.*, visual and orientation features, are all compressed and pooled into this unstructured vector. Therefore, the *structured object-level environment layout* information is discarded during this process. However, the environment layout clues are crucial for the embodied VLN task. As shown in Figure 1(a), to fulfill the instruction like '*go inside the door opposite the bathtub*', the agent of previous methods is confused at step t since the landmark object '*bathtub*' can not be observed, and the agent needs to utilise the historical layout clues (*i.e.*, the door is opposite the bathtub) at step $t-1$ to make the right action.

Therefore, we aim to improve the VLN paradigm via maintaining a structured navigation state, in which three important factors need to be considered. (i) How to represent the navigation state to contain structured layout memory. (ii) How to mine the pivotal layout information for the current and future decisions according to the instruction. (iii) How to store and update a structured state while satisfying the property of long short-term memory.

To achieve the aforementioned objectives, we propose a Structured state-Evolution (SEvol) model as shown in Figure 1(b), where multi-fold innovations are made. (i) Instead of the *vector-based* feature, we propose to adopt the *graph-based* feature as the navigation state, which is capable of holding a structured layout memory. (ii) We design a Reinforced Layout clues Miner (RLM) to mine the most crucial layout information. RLM learns to detect and sample the essential subgraph from the whole layout graph, conditioned on both the current navigation state and the instruction, *e.g.*, sampling subgraph $<$*door*, *opposite*, *bathtub*$>$ according to the language '*go inside the door opposite the bathtub*'. In RLM, we customise a reinforcement learning strategy to make the miner focus on both the immediate interests and the long-term influence of the subgraph sampling. (iii) To effectively store and update a structured graph-based state during the whole navigation process, we devise a Structured Evolving Module (SEM). Specifically, SEM takes the current graph features from RLM as input to evolve the navigation state at each step. The evolution of the navigation state is achieved by interacting with a learnable matrix D (shown in Figure 1(b)), which stores the structured layout memory. D is updated through a matrix version of recurrent neural network. Thus the navigation state contains object-level spatial-temporal relationship that assists the action decision, *e.g.*, the relation '*door opposite bathtub*' in Figure 1(b). Experiments on Room-to-Room (R2R) [2] and Room-for-Room (R4R) show that the proposed SEvol model improves VLN models' performance by large margins.

To summarise, we make the following contributions:

- We propose a simple yet effective SEvol model that provides new insights to the VLN community. The structured navigation state is leveraged to maintain the object-level environment layout during navigation. SEvol achieves state-of-the-art performance on R2R.
- We design a Reinforced Layout clues Miner (RLM) to learn how to detect and sample the most critical subgraph features from the layout graph for current and future action decisions.
- We devise a Structured Evolving Module (SEM) to gradually evolve the structured navigation-state along with the navigation process, maintaining a long short-term layout memory.

2. Related Work

Vision-and-Language Navigation (VLN). Learning to conduct navigation in a simulated photo-realistic environment following the human-annotated nature language instruction, *i.e.*, VLN task [2, 18], has drawn significant interests from both academic and industry fields in recent years. Numerous methods [4, 8, 12, 14, 15, 21, 22, 29, 34, 37–39, 47] have been proposed for the VLN task. Early work Speaker-Follower [8] designs an instructions argumentation strategy, and EnvDrop [34] increases the visual diversity of the synthesised training samples. Besides, [21, 47] introduce auxiliary losses to further improve the ability of cross-modal understanding, and [7, 37] employ trajectory-graph to keep the global navigation memory.

Recently, most VLN works focus on how to utilise a more powerful transformer-based vision-language model to improve performance. CKR [9] adopts the transformer decoder to model the sequential navigation process. VLN-BERT [23] and Airbert [11] leverage the vision-language transformer pre-trained on other large-scale vision-language datasets [32] to conduct instruction-trajectory matching. Other works [14, 39] focus on customising a transformer-based model, which is specifically tailored for the VLN task and can inherit the capability from pre-training models. However, the previous VLN methods pay less attention to the issue of *unstructured navigation state*, which unintentionally reduces the crucial navigation clues, *i.e.*, object-level environment layout.

Dynamic Graph Neural Networks. In the field of data mining, the dynamic graph neural network is an effective tool to mine the information from a sequence of graph-based data that changes over time (*e.g.*, social networks). Some approaches [28, 45] are designed to extract the spatial-temporal relationship from the dynamic graph. [10, 35] take the dynamic graph as the continuous changes over the initial graph, which are advantageous for the event time prediction. Besides, [7, 44] take several frames of the graph and leverage GCN [19]/sequential model [5] to extract features from the graph that changes fast over time. Inspired by the area of data mining, we propose SEM to maintain

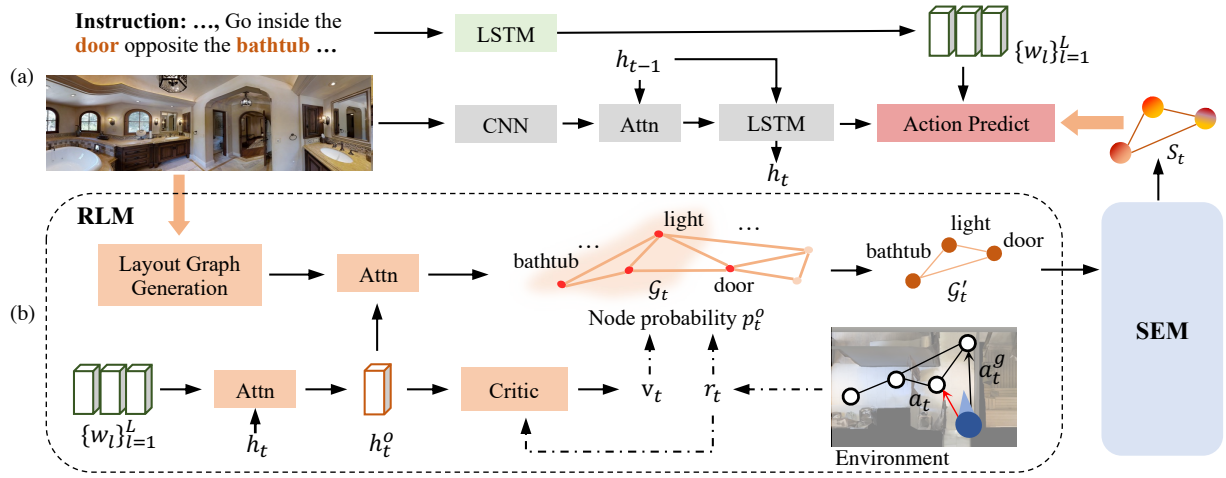

Figure 2. **The Overall Pipeline.** (a) The typical paradigm of VLN methods. (b) The proposed SEvol contains two components, *i.e.*, RLM and SEM. RLM digs out the pivotal layout graph according to the instruction via a customised reinforcement learning strategy. SEM evolves the structured navigation state to maintain the crucial layout memory during navigation.

and update a graph-based navigation state, where different object-level layouts can be extracted dynamically.

3. Method

3.1. Problem Setup and Overview

Problem Setup. In VLN [2], the agent is required to reach the described locations following the instruction. At each step t, the agent observes a panoramic RGB view, which is further divided into 36 discrete views $\{v_{t,i}\}_{i=1}^{36}$. Each view is represented via a RGB image $v_{t,i}$ with its orientation information (heading $\theta_{t,i}$, elevation $\psi_{t,i}$), where i is the view index. Generally, each view's feature $f_{t,i}$ is obtained through $f_{t,i} = [\mathcal{F}(v_{t,i}), \mathcal{E}(\theta_{t,i}, \psi_{t,i})]^\top$, where $[\cdot,\cdot]$ denotes concatenation, $\mathcal{F}(\cdot)$ is the image feature extractor and $\mathcal{E}(\cdot)$ represents the orientation embedding function [34], which is defined as $[cos(\theta_{t,i}), sin(\theta_{t,i}), cos(\psi_{t,i}), sin(\psi_{t,i})]$. Besides, there are K_t candidate views at each step t that are navigable. The agent needs to take an action a_t, *i.e.*, selecting one from K_t candidates to move to that location.

Overview. The basic pipeline of VLN methods [1, 34] is shown in Figure 2(a). The proposed SEvol (shown in Figure 2(b)) is set as an additional branch for providing a structured navigation state. SEvol consists of two components, *i.e.*, Reinforced Layout clues Miner (RLM) and Structured Evolving Module (SEM).

The basic VLN pipeline (Figure 2(a)) leverages a bi-directional LSTM [16] to encode the instruction, and obtains the word-level language feature $\{w_l\}_{l=1}^L$, where L is length. w_L is the sentence level feature and serves as the initial hidden state of the LSTM decoder. At each navigation step t, current visual feature $\{f_{t,i}\}_{i=1}^{36}$ is fed to the LSTM decoder to update its hidden state $h_t \in \mathbb{R}^{N_h \times 1}$:

$$\begin{cases} \tilde{f}_t = attn(h_{t-1}, \{f_{t,i}\}_{i=1}^{36}); \\ h_t = LSTM([\tilde{f}_t, a_{t-1}], h_{t-1}). \end{cases} \quad (1)$$

Note that we represent the normal attention mechanism as $attn(\cdot,\cdot)$ in the paper. For example, $attn(x, Y) = softmax(x^\top W_a Y^\top)Y$, where $x \in \mathbb{R}^{N_1 \times 1}$, $Y \in \mathbb{R}^{N_2 \times N_3}$, and $W_a \in \mathbb{R}^{N_1 \times N_3}$ are trainable parameters.

At each step t, SEvol firstly produces layout graph $\mathcal{G}_t$ from the visual observation. As shown in Figure 2(a), then RLM aims to utilise the language feature $\{w_l\}_{l=1}^L$ (*e.g.*, '*go inside the door opposite the bathtub*') to mine the key sub-graph $\mathcal{G}_t'$ (*e.g.*, '*bathtub*' and '*door*') from $\mathcal{G}_t$. The whole process is optimised through a customised reinforced learning strategy since the RL-based objective can make the miner consider both current and future influences. The reward r_t for the RLM is based on whether the agent moves close or reaches the target position. Next, SEM takes current $\mathcal{G}_t'$ from RLM and layout memory D_{t-1} as inputs to evolve the structured navigation state S_t, where D_{t-1} is an iterable matrix that is used to record the historical layout memory. The evolved S_t is further adopted to predict the final action a_t. In the following, we detailly introduce RLM in Section 3.2 and SEM in Section 3.3, respectively.

3.2. Reinforced Layout clues Miner

Layout Graph Generation. To extract the object-level environment layout, at each step t, we detect top K salient objects $\mathcal{O}_t = \{\mathbf{o}_{t,k}\}_{k=1}^K$ via the Faster R-CNN [30], where $\mathbf{o}_{t,k}$ represents the object entity. We generate a fully connected layout graph $\mathcal{G}_t = \{\mathcal{O}_t, \mathcal{A}_t\}$ base on the object set, where $\mathcal{A}_t$ is the edge set of $\mathcal{G}_t$. To encode the object node set $\mathcal{O}_t$ into a node feature matrix O_t, we consider the semantic and relation position information. The k-th object

feature $O_{t,k}$ is defined as:

$$O_{t,k} = [G(\mathbf{o}_{t,k}), \mathcal{E}(\theta_{\mathbf{o}_{t,k}}, \psi_{\mathbf{o}_{t,k}})]^\top, \tag{2}$$

where $G(\cdot)$ donates the GloVe [27] embedding of the object's label. $\theta_{\mathbf{o}_{t,i}}$ and $\psi_{\mathbf{o}_{t,i}}$ are the heading and elevation of the object relative to the current direction of agent. $\mathcal{E}(\cdot)$ is the orientation embedding function defined in Section 3.1. Then based on the spatial relationships between objects, we define the adjacency matrix A_t as:

$$A_{t,[i,j]} = d(\theta_{\mathbf{o}_{t,i}} - \theta_{\mathbf{o}_{t,j}}), \tag{3}$$

where $d(\cdot)$ is a decrease function of the heading difference and is defined in Section 4.1.

Subgraph Mining. The generated layout graph $\mathcal{G}_t$ contains noise (*e.g.*, navigation-irrelevant object relations). Therefore, to focus on the most pivotal layout information, we propose the RLM to sample a representative subgraph containing M object nodes, as shown in Figure 2(b). Intuitively, the mining process should be dependent on the instruction. Thus we leverage the language-aware feature $\tilde{h}_t = attn(h_t, \{w_l\}_{l=1}^{L})$ to compute the object importance for the subgraph mining:

$$p_t^o = softmax(h_t^{o\top} W_2 O_t^\top), \quad h_t^o = \delta(W_1 \tilde{h}_t), \tag{4}$$

where $W_1 \in \mathbb{R}^{N \times N_h}$ and $W_2 \in \mathbb{R}^{N \times N_o}$ are learnable parameters. $N = 100$ denotes the hidden size and $\delta(\cdot)$ is the ReLU activation function. $p_t^o \in \mathbb{R}^{1 \times K}$ indicates the object's importance during navigation process. Sampling the subgraph $\mathcal{G}_t' = \{\mathcal{O}_t', \mathcal{A}_t'\}$ from $\mathcal{G}_t$ includes two steps. *Firstly*, we select the top M object nodes according to p_t^o. Thus we get the sub set of $\mathcal{O}_t$: $\mathcal{O}_t' = \{\mathbf{o}_{t,m}\}_{m=1}^{M}$ and the importance scores of the selected objects $p_t^{o\prime} \in \mathbb{R}^{1 \times M}$. *Secondly*, we obtain the edge set $\mathcal{A}_t'$ via keeping the corresponding edges between selected objects:

$$\mathcal{A}_t' = \{e_{t,ij} | \mathbf{o}_{t,i} \in \mathcal{O}_t' \wedge \mathbf{o}_{t,j} \in \mathcal{O}_t' \wedge e_{t,ij} \in \mathcal{A}_t\}. \tag{5}$$

Therefore, the node feature and the adjacency matrix of M-order graph $\mathcal{G}_t'$ are $O_t' \in \mathbb{R}^{M \times N_o}$ and $A_t' \in \mathbb{R}^{M \times M}$.

Since navigation is a long-term planning task, RLM should also consider future navigation decisions. Thus we customise a reinforcement learning strategy to guide the subgraph sampling process. The reward for the object's node selection is based on the navigation action feedback from environments as shown in Figure 2(b). At each step t, the agent's action reward is defined as: (i) If the agent's distance from destination decreases, the reward r_t will be 1. Otherwise, it will be -1. (ii) When the agent successfully finishes the navigation task, it will gain a large reward $r_T = 4$, and if the agent stops at the wrong position, it will get a negative reward value $r_T = -2$. The navigation reward can be seen as the reward for the RLM. At each step t,

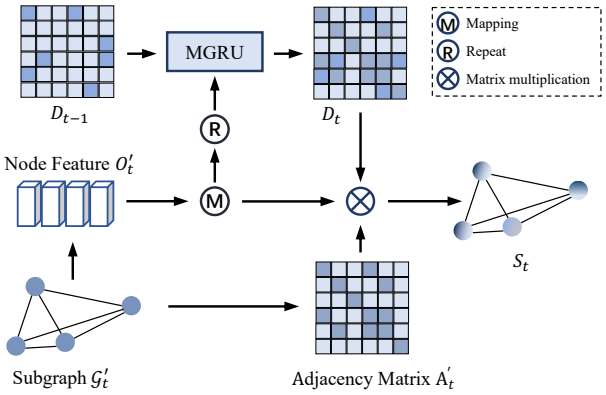

Figure 3. Illustration of our SEM module. SEM takes the layout graph $\mathcal{G}_t'$ from RLM to update the layout memory D_t and generate the structured navigation state S_t.

the critic network $critic(\cdot)$ estimates the state value v_t, and each action reward r_t for RLM is:

$$r_t = \gamma^{(T-t)} v_T + r_t, \quad v_t = critic(h_t^o), \tag{6}$$

where γ is the decay rate, T denotes the total steps of the trajectory. Therefore, the advantage of each action reward against state value is $\varsigma_t = r_t - v_t$. The action of subgraph miner is to choose M objects from $\mathcal{O}_t$, and the objective functions based on A2C [24] is formulated as:

$$\begin{cases} \mathcal{L}_{sa} = \sum_{t=0}^{T} \sum_{m=0}^{M} -\varsigma_t \log p_{t,m}^{o\prime}; \\ \mathcal{L}_{sc} = \sum_{t=0}^{T} \varsigma_t^2; \\ \mathcal{L}_{sd} = \sum_{t=0}^{T} \sum_{k=0}^{K} -p_{t,k}^o \log p_{t,k}^o; \\ \mathcal{L}_s = \lambda_1 \mathcal{L}_{sa} + \lambda_2 \mathcal{L}_{sc} + \lambda_3 \mathcal{L}_{sd}, \end{cases} \tag{7}$$

where $\mathcal{L}_{sa}$ optimises the object selection, $\mathcal{L}_{sc}$ optimises the critic model, and $\mathcal{L}_{sd}$ aims to avoid the object importance scores p_t^o degeneration to uniform distribution. $\lambda_i (i = 1, 2, 3)$ is the weighting factor that controls the relative importance of each term.

3.3. Structured Evolving Module

The object-level layout information is contained in the generated layout graph. However, the navigation is a sequential process, where the layout graph dynamically changes over time. Thue we propose the Structured Evolving Module (SEM) to dynamically handle the layout graph $\mathcal{G}_t'$, to update the navigation state S_t at each step t. As shown in Figure 3, we adopt a learnable matrix D_t to record the structured layout memory. We leverage the matrix version of GRU (MGRU) [26] to handle $D_t \in \mathbb{R}^{N \times N}$, where

$N = 100$ is a hyperparameter. The D_t is updated by the node feature O'_t of the subgraph from RLM. We first repeat and mapping $O'_t \in \mathbb{R}^{M \times N_o}$ to produce a square matrix $Q_t \in \mathbb{R}^{N \times N}$ $(M < N)$. Then the updating process of $D_{t-1} \rightarrow D_t$ can be formulated as:

$$\begin{cases} Z_t = \sigma(W_z Q_t + U_z D_{t-1} + B_z); \\ R_t = \sigma(W_r Q_t + U_r D_{t-1} + B_r); \\ \tilde{D}_t = \tanh(W_h Q_t + U_h(R_t D_{t-1}) + B_h); \\ D_t = (1 - Z_t) \circ D_{t-1} + Z_t \circ \tilde{D}_t, \end{cases} \tag{8}$$

where $\circ$ denotes the Hadamard product, σ represents the sigmoid function and the trainable parameters are all square matrix of order N. For the initialisation of D_t, since there is no history layout information at the first step, the D_0 should condition on the language instruction:

$$D_0 = \sigma(I_0 w_L^\top W_3 + B_i), \tag{9}$$

where $I_0 \in \mathbb{R}^{N \times 1}$, $B_i \in \mathbb{R}^{N \times N}$,$W_3 \in \mathbb{R}^{N_h \times N}$ are trainable parameters. $w_L \in \mathbb{R}^{N_h \times 1}$ is the sentence level feature.

The obtained history structure memory D_t guides a graph convolution on the current layout graph to produce the current structured navigation state:

$$S_t = \tanh(A'_t \sigma(O'_t W_m) D_t), \tag{10}$$

where $W_m \in \mathbb{R}^{N_o \times N}$ is learnable. Due to the interaction between the subgraph $\mathcal{G}'_t$ from RLM and the layout memory matrix D_t, the S_t carries the layout relation of the current step with historical memory. We use the language context-aware feature $\tilde{h}_t$ to conduct attention pooling on the S_t to build a global representation h_t^s for action selection:

$$h_t^s = \sigma(W_4[\tilde{s}_t, h_t]), \quad \tilde{s}_t = attn(\tilde{h}_t, S_t), \tag{11}$$

where $W_4 \in \mathbb{R}^{(N_h + N_o) \times N_h}$ is learnable.

Finally, h_t^s serves as the action selector in the action prediction module. For simplicity, the action prediction module computes the attention similarity between the h_t^s and the feature $f_t = [f_{t,i}]_{i=0}^{K_t}$ that represents the candidate views:

$$p_t = softmax(h_t^s W_5 f_t), \tag{12}$$

where $W_5 \in \mathbb{R}^{N_h \times N_v}$ is the trainable parameter, N_v is the dimension of visual feature, and $f_{t,0}$ is a zero vector representing the stop action. p_t is the action probability distribution at step t. The agent samples/selects a candidate view based on p_t and move to the viewpoint corresponding to that view.

3.4. Training Loss

Our training objective is composed of three different parts: the imitation learning loss $\mathcal{L}_I$, the reinforcement learning for action prediction $\mathcal{L}_r$ and the subgraph extraction loss $\mathcal{L}_s$ mentioned before. During the training process, we implement the student-force training strategy, in which the agent takes action that is predicted by itself. With the a_t^g as the shortest path to the destination, the imitation learning loss is defined as:

$$\mathcal{L}_I = \lambda \sum_{t=0}^{T} a_t^g \log P_t. \tag{13}$$

The reinforcement learning uses the A2C [24] strategy:

$$\mathcal{L}_r = \sum_{t=0}^{T} -a_t \log(P_t) adv_t, \tag{14}$$

where the adv_t is the advantage at step t in A2C algorithm. The overall training loss can be defined as:

$$\mathcal{L} = \mathcal{L}_s + \lambda_I \mathcal{L}_I + \lambda_r \mathcal{L}_r, \tag{15}$$

where λ_I, λ_r are coefficients for balancing the loss terms.

4. Experiments

4.1. Experimental Setup

Datasets. We adopt Room-to-Room(R2R) dataset [2] and Room-for-Room dataset [18] to validate our method. In the R2R dataset, there are $7,189$ trajectories and three natural language instructions describe each. It is divided into train, validation seen (val-seen), validation unseen (val-unseen), and test sets. The environments of val-unseen and test are not in the train set. The R4R dataset contains longer and twistier paths, which is the extended version of R2R. The R4R dataset is divided into train, validation seen and validation unseen sets. The R2R and R4R are built on the Matterport3D simulator [2, 3], which consists of $10,567$ panoramic views in 90 real word indoor environments.

Evaluation Metrics. To compare with the existing methods, we report the commonly used evaluation metrics on the R2R dataset, *i.e.*, Trajectory Length (TL), Navigation Error (NE), Success Rate (SR), and Success Rate weighted by Path Length (SPL). Following [1,17,18] three more evaluations are used for R4R, *i.e.*, normalized Dynamic Time Warping (nDTW), Success rate weighted normalized Dynamic Time Warping (sDTW) and Coverage weighted by Length Score (CLS).

Implementation Details. We use a 2-layer bi-directional LSTM with a cell size of 256 in each direction for the language encoder and use GloVe [27] for word embedding. We use the CLIP-ViT-B-32 as our visual encoder. The LSTM decoder is with a cell size of 512. The SEvol can be seen as an additional branch for the LSTM based VLN model. More specifically, h_t^s produced by SEvol can replace h_t, the output of the LSTM. At each step, we detect 400 objects to build $\mathcal{G}_t$. The object detector is Faster R-CNN [30]

Agent	Val Seen				Val Unseen				Test Unseen			
	TL	NE↓	SR↑	SPL↑	TL	NE↓	SR↑	SPL↑	TL	NE↓	SR↑	SPL↑
Random	9.58	9.45	0.16	-	9.77	9.23	0.16	-	9.89	9.79	0.13	0.12
Human	-	-	-	-	-	-	-	-	11.85	1.61	0.86	0.76
Pretrain Model:												
PRESS [43]	10.57	4.39	0.58	0.55	10.36	5.28	0.49	0.45	10.77	5.49	0.49	0.45
PREVALENT [12]	10.32	3.67	0.69	0.65	10.19	4.71	0.58	0.53	10.51	5.30	0.54	0.51
VLNBERT (init. OSCAR) [14]	10.79	3.11	0.71	0.67	11.86	4.29	0.59	0.53	12.34	4.59	0.57	0.53
VLNBERT (init. PREVALENT) [14]	11.13	2.90	0.72	0.68	12.01	3.93	0.63	0.57	12.35	4.09	0.63	0.57
VLNBERT+REM [4]	10.88	2.48	75.4	71.8	12.44	3.89	0.64	0.60	13.11	3.87	0.65	0.59
ORIST (init. UNITER) [39]	-	-	-	-	10.90	4.72	0.57	0.51	11.31	5.10	0.57	0.52
w/ Data Augmentation:												
Speaker-Follower [8]	-	3.36	0.66	-	-	6.62	0.35	-	14.82	6.62	0.35	0.28
SM [21]	-	3.22	0.67	0.58	-	5.52	0.45	0.32	18.04	5.67	0.48	0.35
RCM+SIL [40]	10.65	3.53	0.67	-	11.46	6.09	0.43	-	11.97	6.12	0.43	0.38
Regretful [22]	-	3.23	0.69	0.63	-	5.32	0.50	0.41	13.69	5.69	0.48	0.40
IL+RL+REM [4]	10.18	4.61	0.58	0.55	**9.40**	5.59	0.48	0.44	**9.81**	5.67	0.48	0.45
SSM [37]	14.70	**3.10**	**0.71**	0.62	20.70	4.32	**0.62**	0.45	20.40	4.57	0.61	0.46
VLNBERT (no init.) [14]	9.78	3.92	0.62	0.59	10.31	5.10	0.50	0.46	11.15	5.45	0.51	0.47
EnvDrop [34]	11.00	3.99	0.62	0.59	10.70	5.22	0.52	0.48	11.66	5.23	0.51	0.47
EnvDrop+REM [4]	11.13	3.14	0.70	0.66	14.84	4.99	0.53	0.48	10.73	5.40	0.54	0.50
AuxRN [47]	-	3.33	0.70	**0.67**	-	5.28	0.55	0.50	-	5.15	0.55	0.51
RelGraph [13]	**10.13**	3.47	0.67	0.65	9.99	4.73	0.57	0.53	10.29	4.75	0.55	0.52
NvEM [1]	11.09	3.44	0.69	0.65	11.83	4.27	0.60	0.55	12.98	4.37	0.58	0.54
EnvDrop+SEvol	12.55	3.70	0.61	0.57	14.67	4.39	0.59	0.53	14.30	3.70	0.59	0.55
NvEM+SEvol	11.97	3.56	0.67	0.63	12.26	**3.99**	**0.62**	**0.57**	13.40	**4.13**	**0.62**	**0.57**
w/o Data Augmentation:												
Student-Forcing [2]	11.33	6.01	0.39	-	8.39	7.81	0.22	-	**8.13**	7.85	0.20	0.18
RPA [41]	**8.46**	5.56	0.43	-	**7.22**	7.65	0.25	-	9.15	7.53	0.25	0.23
Regretful [22]	-	3.69	0.65	**0.59**	-	5.36	0.48	0.37	-	-	-	-
EGP [7]	-	-	-	-	-	5.34	0.52	0.41	-	-	-	-
EnvDrop [34]	10.10	4.71	0.55	0.53	9.37	5.49	0.47	0.43	-	-	-	-
Active Perception [38]	19.80	**3.35**	**0.66**	0.51	19.90	4.40	0.55	0.40	21.0	4.77	0.56	0.37
SSM [37]	13.50	3.77	0.65	0.57	18.90	4.88	0.56	0.42	18.50	4.66	0.57	0.44
EnvDrop+SEvol	12.33	4.19	0.59	0.54	13.59	4.72	0.55	0.49	15.48	4.52	0.57	0.52
NvEM+SEvol	12.07	3.96	0.63	**0.59**	12.45	**4.15**	**0.61**	**0.55**	13.10	**4.25**	**0.60**	**0.55**

Table 1. **Comparisons on R2R dataset**. Comparison of single run performance with the state-of-the-art methods on R2R. The proposed SEvol boosts the performance in terms of all the key metrics on the unseen environments.

Agent	Val Seen						Val Unseen					
	NE↓	SR↑	SPL↑	CLS↑	nDTW↑	sDTW↑	NE↓	SR↑	SPL↑	CLS↑	nDTW↑	sDTW↑
EnvDrop [34]	-	0.52	0.41	0.53	-	0.27	-	0.29	0.18	0.34	-	0.09
RCM-b [36]	-	-	-	-	-	-	-	0.29	0.21	0.35	0.30	0.13
OAAM [46]	-	**0.56**	0.49	**0.54**	-	0.32	-	0.31	0.23	0.40	-	0.11
RelGraph [13]	**5.14**	0.55	**0.50**	0.51	0.48	**0.35**	7.55	0.35	0.25	0.37	0.32	0.18
NvEM [1]	5.38	0.54	0.47	0.51	**0.48**	**0.35**	**6.85**	0.38	0.28	**0.41**	**0.36**	**0.20**
NvEM+SEvol	5.77	0.50	0.40	0.48	0.45	0.30	6.90	**0.39**	**0.29**	**0.41**	**0.36**	**0.20**

Table 2. **Comparisons on R4R dataset**. Comparison of single run performance with the state-of-the-art methods on R4R.

trained on Visual Genome Dataset [20] which classifies the 100 most frequent objects appearing in the instructions and environments. The REM module selects 5 objects from $\mathcal{G}_t$ to induces the subgraph $\mathcal{G}'_t$. The edge weight function over the objects' angle difference x is $d(x) = \frac{\pi}{\pi+6x}$. Following [34], we apply the two-stage training strategy on R2R

Name	SEM	RLM	Aug_{bp}	Val-Seen				Val-Unseen			
				SR↑	NE↓	TL↓	SPL↑	SR↑	NE↓	TL↓	SPL↑
NvEM [1]*				0.61	4.25	**10.85**	0.58	0.57	4.62	**11.24**	0.51
#1	✓			0.62	3.79	12.16	0.58	0.60	4.31	12.41	0.54
#2	✓	✓		0.63	3.96	12.07	0.59	0.61	4.15	12.45	0.55
#3	✓		✓	0.64	3.79	11.66	0.59	0.61	4.14	12.40	0.56
#4	✓	✓	✓	**0.67**	**3.56**	11.97	**0.63**	**0.62**	**3.99**	12.26	**0.57**

Table 3. **Ablations.** The performance is gradually improved with the continuous addition of proposed modules, especially on val-unseen. The reproduction result of NvEM [1] *w/o data augmentation* is shown at the first line. Experiments confirm the effectiveness of RLM/SEM modules under both the *w/* and *w/o data augmentation* setup. Note that Aug_{bp} denotes data augmentation.

Dim(D_t)	Val-Seen			Val-Unseen		
	SR↑	NE↓	SPL↑	SR↑	NE↓	SPL↑
–	0.61	4.25	0.58	0.57	4.62	0.51
50	**0.65**	**3.72**	**0.60**	0.58	4.43	0.52
100	0.63	3.96	0.59	**0.61**	**4.15**	**0.55**
200	0.63	3.89	0.58	0.60	4.33	0.54

(a) **Capacity of Layout Memory.** The Dimension of D_t in SEM. Model with $dim(D_t) = 100$ performs best on val-unseen.

Num	Val-Seen				Val-Unseen			
	SR↑	NE↓	TL↓	SPL↑	SR↑	NE↓	TL↓	SPL↑
0	0.61	4.25	**10.85**	0.58	0.57	4.62	**11.24**	0.51
3	0.60	4.21	11.26	0.56	0.57	4.45	11.67	0.52
5	**0.63**	**3.96**	12.07	**0.59**	**0.61**	**4.15**	12.45	**0.55**
20	0.59	4.03	12.28	0.55	0.59	4.33	12.64	0.54

(b) **Subgraph Size.** The size of subgraph selected by RLM. The model performs best on val-unseen when the subgraph with size of 5.

Table 4. **Ablations.** We mainly focus on the key metrics for each ablation setting. Note that the results in this table are obtained under the *w/o data augmentation* setting.

$m(x)$	Val-Seen			Val-Unseen		
	SR↑	NE↓	SPL↑	SR↑	NE↓	SPL↑
x	**0.63**	3.96	**0.59**	**0.61**	4.15	**0.55**
e^x	0.59	4.26	0.55	0.57	4.53	0.53
1	0.61	**3.88**	0.57	0.58	**3.34**	0.53

Table 5. Performances with different edge weight generation functions $d(m(x))$, where x is heading difference between two objects.

dataset. At the first stage, we train the agent on the train set. At the second stage, the agent is trained with the back translation based data augmentation method [8] The learning rate is 10^{-4}. The weighting factor's values are $\lambda_1 = 0.2$, $\lambda_2 = 0.1$, $\lambda_3 = 0.01$, $\lambda_I = 0.2$ and $\lambda_r = 1$. The batch size is 64. We train our model for $8,000$ iterations for stage 1, and $20,000$ iterations for stage 2 and select the model with the best performance on val-unseen set.

4.2. Comparison with State-of-the-art Methods

R2R Dataset. As shown in Table 1, under the *w/ data argumentation* setting, our best model reach the same SR with the previous best one (SSM [37]), but we achieve a significant improvement in SPL (from 0.45 to 0.57) on val-unseen. The NvEM+SEvol surpasses NvEM [1] with $+2\%$ in both SR and SPL. Besides, we achieve $+4\%$ and $+3\%$ absolute improvement in SR and SPL than NvEM [1] on test set. SSM [37] sacrifices the navigation efficiency to achieve 0.61 SR on test set, and we achieve higher SR ($+1\%$) and higher efficiency ($+11\%$ in SPL). The SEvol also boosts the performance of EnvDrop for a large margin. Under the *w/o data argumentation* setting, as shown in Table 1, we achieve the best performance in terms of SR and SPL on both val-unseen and test sets, and it is higher than the previous best methods in a large margin. Even compared with the previous state-of-the-art NvEM [1] trained with augmented data, the NvEM+SEvol performs better on both val-unseen and test set in terms of SR and SPL.

R4R Dataset. As shown in Table 2, we compare the proposed method with the current state-of-the-art methods on the R4R benchmark. The NvEM+SEvol achieves the best performance on most of the metrics on val-unseen set. And it suppresses the NvEM model on both SR and SPL by 1% and achieves the same accuracy on the trajectory fidelity metrics. The experiments show that the proposed SEvol also works on samples with longer navigation trajectories.

4.3. Ablation Study

In this section, we evaluate the effectiveness of the key components of our model, *i.e.*, RLM and SEM. Note that the base model in our experiments is NvEM [1]. We reproduce the experiment without data argumentation and add the key components of our model step by step.

Structured Evolving Module (SEM). As shown in Table 3, compare with base agent, '#1' with SEM lifts SR and SPL from (0.57, 0.51) to (0.60, 0.54) on val-unseen. It confirms that the layout memory and structured state are conducive to navigation. Moreover, after training with augmented data ('#3'), the model still benefits from structured

state maintaining.

Reinforced Layout clues Miner (RLM). As shown in Table 3, comparing to '#1', '#2' confirms the effectiveness of RLM by boosting SR and SPL from 0.60, 0.54 to 0.61, 0.65. The advantage remains after training with augmentation data, as shown in '#4'. It proves that selecting suitable objects will help the layout memory to keep useful information of the environment and produce better structured navigation state.

Capacity of Structure Memory. As shown in Table 4(a), we investigate how the capacity of the layout memory in SEM impacts the performance by changing the dimension of D_t. Note that $dim = 0$ represents the base model without the SEvol. We can observe that no matter the dimension of D_t, the model with SEM performs better than the baseline model. When the dimension rises from 50 to 100, SR on val-unseen increases to 0.61 from 0.58, which illustrates that the larger capacity of the structure memory has better expressiveness. The model performs best when $dim = 100$. SR on val-unseen decreases to 0.60 when $dim = 200$, which indicates that high-dimensional D_t increases the convergence difficulty.

Subgraph Size. As shown in Table 4(b), we investigate how the node number of $\mathcal{G}'_t$ selected by RLM influences the performance. We compare the models with different sizes of subgraph, *i.e.*, node number of 3, 5 and 20. The model with the subgraph $\mathcal{G}'_t$ size of 5 achieves the best performance. When the subgraph size increases to 20, SR on val-unseen decreases, which indicates that a large layout graph brings more noise.

Layout Graph Construction. We investigate the different spatial layout graph construction methods. Specifically, we choose mapping functions $m(\cdot)$ to influence the decay speed of edge weights over x, the difference of angle between objects. As shown in Table 5, x represents the angle difference between two objects. When $m(x) = x$, the model has best performance, which is better than $m(x) = 1$. When $m(x) = e^x$, the model has the lowest SR on val-unseen. Thus the edge weight should not decrease too fast since the relationships among objects in the different directions is useful for navigation.

4.4. Visualisation

To demonstrate that the history layout information maintained by the SEvol model assists the agent's decision-making process, we visualise the trajectories generated by the agent with or without the SEvol model. In Figure 4, the agent with SEvol does not stop at the first exit sign, then gets to the right destination. However, the agent without SEvol stops at the wrong position. Due to the lack of the object layout history information, it can not distinguish the first and the second sign.

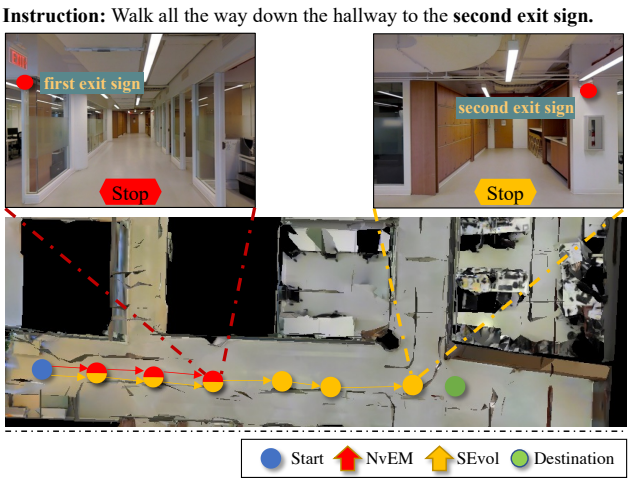

Figure 4. The decision divergence between the SEvol and baseline model NvEM [1]. The proposed SEvol obtains more layout clues than the baseline model.

5. Conclusion and Discussion

The object-level layout information is critical for the Vision-and-Language Navigation task. To utilise the layout information, we propose a novel model for vision-and-language navigation, named the Structured state-Evolution model (SEvol), which maintains a structured memory and evolves a structured navigation state. We offer RLM to extract the navigation-related subgraph from the environment and SEM to maintain a layout memory for evolving structured navigation. Extensive experiments demonstrate the effectiveness of our proposed methods. The proposed SEvol substantially improves VLN performance on the R2R dataset, proving the importance of layout information in the VLN task. We believe that this work will bring new insights to the research around vision-and-language navigation.

Limitations. Although SEvol outperforms previous state-of-the-art methods by a large margin, the layout information in SEvol only contains orientation relations between objects. Thus how to represent the layout relation containing more valuable clues is worth studying in the future. Besides, due to the insufficient environment data on the R2R dataset, models trained on R2R may have bias over some typical environments. Thus constructing more diverse datasets is a primary task for the VLN community. In the future, we will continue the research on the VLN task around the limitations.

Acknowledgements. This research is partly supported by National Natural Science Foundation of China (Grant 62122010, 61876177), Fundamental Research Funds for the Central Universities, and Key R&D Program of Zhejiang Province (2022C01082).

References

[1] Dong An, Yuankai Qi, Yan Huang, Qi Wu, Liang Wang, and Tieniu Tan. Neighbor-view enhanced model for vision and language navigation. In *ACM MM*, 2021. 3, 5, 6, 7, 8

[2] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *CVPR*, 2018. 1, 2, 3, 5, 6

[3] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *arXiv preprint arXiv:1709.06158*, 2017. 1, 5

[4] Liu Chong, Zhu Fengda, Chang Xiaojun, Liang Xiaodan, Ge Zongyuan, and Shen Yi-Dong. Vision-language navigation with random environmental mixup. In *ICCV*, 2021. 2, 6

[5] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014. 2

[6] Matt Deitke, Winson Han, Alvaro Herrasti, Aniruddha Kembhavi, Eric Kolve, Roozbeh Mottaghi, Jordi Salvador, Dustin Schwenk, Eli VanderBilt, Matthew Wallingford, Luca Weihs, Mark Yatskar, and Ali Farhadi. RoboTHOR: An Open Simulation-to-Real Embodied AI Platform. In *CVPR*, 2020. 1

[7] Zhiwei Deng, Karthik Narasimhan, and Olga Russakovsky. Evolving graphical planner: Contextual global planning for vision-and-language navigation. In *NeurIPS*, 2020. 2, 6

[8] Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. Speaker-follower models for vision-and-language navigation. In *NeurIPS*, 2018. 1, 2, 6, 7

[9] Chen Gao, Jinyu Chen, Si Liu, Luting Wang, Qiong Zhang, and Qi Wu. Room-and-object aware knowledge reasoning for remote embodied referring expression. In *CVPR*, 2021. 2

[10] Palash Goyal, Sujit Rokka Chhetri, and Arquimedes Canedo. dyngraph2vec: Capturing network dynamics using dynamic graph representation learning. *KBS*, 2020. 2

[11] Pierre-Louis Guhur, Makarand Tapaswi, Shizhe Chen, Ivan Laptev, and Cordelia Schmid. Airbert: In-domain pretraining for vision-and-language navigation. In *ICCV*, 2021. 2

[12] Weituo Hao, Chunyuan Li, Xiujun Li, Lawrence Carin, and Jianfeng Gao. Towards learning a generic agent for vision-and-language navigation via pre-training. In *CVPR*, 2020. 2, 6

[13] Yicong Hong, Cristian Rodriguez-Opazo, Yuankai Qi, Qi Wu, and Stephen Gould. Language and visual entity relationship graph for agent navigation. In *NeurIPS*, 2020. 6

[14] Yicong Hong, Qi Wu, Yuankai Qi, Cristian Rodriguez-Opazo, and Stephen Gould. A recurrent vision-and-language bert for navigation. In *CVPR*, 2021. 2, 6

[15] Ronghang Hu, Daniel Fried, Anna Rohrbach, Dan Klein, Trevor Darrell, and Kate Saenko. Are you looking? grounding to multiple modalities in vision-and-language navigation. In *ACL*, 2019. 2

[16] Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*, 2015. 3

[17] Gabriel Ilharco, Vihan Jain, Alexander Ku, Eugene Ie, and Jason Baldridge. General evaluation for instruction conditioned navigation using dynamic time warping. In *NeurIPS*, 2019. 5

[18] Vihan Jain, Gabriel Magalhaes, Alexander Ku, Ashish Vaswani, Eugene Ie, and Jason Baldridge. Stay on the Path: Instruction Fidelity in Vision-and-Language Navigation. In *ACL*, 2019. 2, 5

[19] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017. 2

[20] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *IJCV*, 2017. 6

[21] Chih-Yao Ma, Jiasen Lu, Zuxuan Wu, Ghassan AlRegib, Zsolt Kira, Richard Socher, and Caiming Xiong. Self-monitoring navigation agent via auxiliary progress estimation. In *ICLR*, 2019. 1, 2, 6

[22] Chih-Yao Ma, Zuxuan Wu, Ghassan AlRegib, Caiming Xiong, and Zsolt Kira. The regretful agent: Heuristic-aided navigation through progress estimation. In *CVPR*, 2019. 1, 2, 6

[23] Arjun Majumdar, Ayush Shrivastava, Stefan Lee, Peter Anderson, Devi Parikh, and Dhruv Batra. Improving vision-and-language navigation with image-text pairs from the web. In *ECCV*, 2020. 2

[24] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *ICML*, 2016. 4, 5

[25] Igor Mordatch and Pieter Abbeel. Emergence of grounded compositional language in multi-agent populations. In *AAAI*, 2018. 1

[26] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao Schardl, and Charles Leiserson. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In *AAAI*, 2020. 4

[27] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, 2014. 4, 5

[28] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *KDD*, 2014. 2

[29] Guhur Pierre-Louis, Makarand Tapaswi, Chen Shizhe, Laptev Ivan, and Schmid1 Cordelia. Airbert: In-domain pretraining for vision-and-language navigation. In *ICCV*, 2021. 2

[30] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015. 3, 5

[31] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A platform for embodied ai research. In *ICCV*, 2019. 1

[32] Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *ACL*, 2018. 2

[33] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks. In *CVPR*, 2020. 1

[34] Hao Tan, Licheng Yu, and Mohit Bansal. Learning to navigate unseen environments: Back translation with environmental dropout. In *NAACL*, 2019. 1, 2, 3, 6

[35] Rakshit Trivedi, Hanjun Dai, Yichen Wang, and Le Song. Know-evolve: Deep temporal reasoning for dynamic knowledge graphs. In *ICML*, 2017. 2

[36] Jain Vihan, Ku Alexander, Ie Eugene, and Baldridge Jason. General evaluation for instruction conditioned navigation using dynamic time warping. In *NeurIPS*, 2019. 6

[37] Hanqing Wang, Wenguan Wang, Wei Liang, Caiming Xiong, and Jianbing Shen. Structured scene memory for vision-language navigation. In *CVPR*, 2021. 2, 6, 7

[38] Hanqing Wang, Wenguan Wang, Tianmin Shu, Wei Liang, and Jianbing Shen. Active visual information gathering for vision-language navigation. In *ECCV*, 2020. 2, 6

[39] Hanqing Wang, Wenguan Wang, Tianmin Shu, Wei Liang, and Jianbing Shen. The road to know-where: An object-and-room informed sequential bert for indoor vision-language navigation. In *ECCV*, 2020. 2, 6

[40] Xin Wang, Qiuyuan Huang, Asli Celikyilmaz, Jianfeng Gao, Dinghan Shen, Yuan-Fang Wang, William Yang Wang, and Lei Zhang. Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *CVPR*, 2019. 6

[41] Xin Wang, Wenhan Xiong, Hongmin Wang, and William Yang Wang. Look before you leap: Bridging model-free and model-based reinforcement learning for planned-ahead vision-and-language navigation. In *ECCV*, 2018. 6

[42] Fei Xia, Amir R. Zamir, Zhi-Yang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson Env: real-world perception for embodied agents. In *CVPR*, 2018. 1

[43] Li Xiujun, Li Chunyuan, Xia Qiaolin, Bisk Yonatan, Asli Celikyilmaz, Gao Jianfeng, A. Smith Noah, and Yejin Choi. Robust navigation with language pretraining and stochastic sampling. In *EMNLP-IJCNLP*, 2019. 6

[44] Hansheng Xue, Luwei Yang, Wen Jiang, Yi Wei, Yi Hu, and Yu Lin. Modeling dynamic heterogeneous network for link prediction using hierarchical attention with temporal rnn. In *ECML*, 2021. 2

[45] Wenchao Yu, Wei Cheng, Charu C Aggarwal, Kai Zhang, Haifeng Chen, and Wei Wang. Netwalk: A flexible deep embedding approach for anomaly detection in dynamic networks. In *KDD*, 2018. 2

[46] Qi Yuankai, Pan Zizheng, Zhang Shengping, den Hengel Anton van, and Wu Qi. Object-and-action aware model for visual language navigation. In *ECCV*, 2020. 6

[47] Fengda Zhu, Yi Zhu, Xiaojun Chang, and Xiaodan Liang. Vision-language navigation with self-supervised auxiliary reasoning tasks. In *CVPR*, 2020. 2, 6