
Zhiwei Deng  Karthik Narasimhan  Olga Russakovsky
Department of Computer Science
Princeton University
{zhiweid, karthikn, olgarus}@cs.princeton.edu

Abstract

The ability to perform effective planning is crucial for building an instruction-following agent. When navigating through a new environment, an agent is challenged with (1) connecting the natural language instructions with its progressively growing knowledge of the world; and (2) performing long-range planning and decision making in the form of effective exploration and error correction. Current methods are still limited on both fronts despite extensive efforts. In this paper, we introduce the Evolving Graphical Planner (EGP), a model that performs global planning for navigation based on raw sensory input. The model dynamically constructs a graphical representation, generalizes the action space to allow for more flexible decision making, and performs efficient planning on a proxy graph representation. We evaluate our model on a challenging Vision-and-Language Navigation (VLN) task with photorealistic images, and achieve superior performance compared to previous navigation architectures. For instance, we achieve a 53% success rate on the test split of the Room-to-Room navigation task [1] through pure imitation learning, outperforming previous navigation architectures by up to 5%.

1 Introduction

Recent work has made remarkable progress towards building autonomous agents that navigate by following instructions [2, 3, 4, 5, 6, 7, 8, 9, 10] and constructing memory structures for maps [11, 12, 13]. An important problem setting within this space is the paradigm of online navigation, where an agent needs to perform navigation based on goal descriptions in an unseen environment using a limited number of steps [14, 1].

In order to successfully navigate through an unseen environment, an agent needs to overcome two key challenges. First, the instructions given to the agent are natural language descriptions of the goal and the landmarks along the way; these descriptions need to be grounded onto the evolving visual world that the agent is observing. Second, the agent needs to perform non-trivial planning over a large action space, including: 1) deciding which step to take next to resolve ambiguities in the instructions through novel observations and 2) gaining a better understanding of the environment layout in order to progress towards the goal or recover from its prior mistakes. Notably this planning requires not only selecting from an increasingly large set of possible actions but also performing complex long-term reasoning.

Existing work tackles only one or two components of the above and may require additional preprocessing steps. Some are constrained to use local control policies [14, 2] or use rule-based algorithms such as beam or $A^*$ search [2, 15, 14] to perform localized path corrections. Others focus on processing long-range observations instead of actions [16] or employ offline pre-training schemes to learn topological structures [12, 13, 17]. This is challenging since accurate construction of graphs is non-trivial and requires special adaptation to work during real-time navigation [13, 17].

In this paper, we propose the Evolving Graphical Planner (EGP) (Figure 1), which 1) dynamically constructs a graphical map of the environment in an online fashion during exploration and 2) incorporates a global planning module for selecting actions. EGP can operate directly on raw sensory inputs in partially observable settings by building a structured representation of the geometric layout of the environment using discrete symbols to represent visited states and unexplored actions. This expressive representation allows our agent to choose from a greater number of global actions conditioned on the text instruction and perform course corrections if needed.

Incorporating a global navigation module is challenging since we do not always have access to ground truth supervisions from the environment. Further, the ever-expanding size of the global graphs requires a scalable action selection module. To solve the first challenge, we introduce a novel method for training our planning modules using imitation learning – this allows the agent to efficiently learn how to select global actions and backtrack when necessary. For the second, we introduce proxy graphs, which are local approximations of the entire map and allow for more scalable planning. Our entire model is end-to-end differentiable under a pure imitation learning framework.

We test EGP on two benchmarks for 3D navigation with instructions – Room-to-Room [1] and Room-for-Room [18]. Our model outperforms several state-of-the-art backbone architectures on both datasets – e.g., on Room-to-Room, we achieve a 5% improvement in Success Rate over the Regretful agent [19]. We also perform a series of ablation studies on our model to justify the design choices. Our implementation is available at https://github.com/Lucas2012/EvolvingGraphicalPlanner.

2 Related work

Embodied Navigation Agent Many recent papers develop neural architectures for navigation tasks [19, 2, 14, 20, 21, 16, 23, 24, 25]. Vision-and-Language Navigation (VLN) [1, 5] is one representative task that focuses on language-driven navigation across photo-realistic 3D environments. Anderson et al. [1] propose the Room-to-Room benchmark and an attention-based sequence-to-sequence method. Fried et al. [2] extend the model by a pragmatic agent with the ability to synthesize data through a speaker model. With an emphasize on grounding, the self-monitoring agent [14] adopts a co-grounding module and a progress estimation auxiliary task for more progress-sensitive alignment. Similarly, an intrinsic reward [26] is introduced to improve cross-modal alignment for navigation agent. Ma et al. extends the existing works towards a graph search like algorithm by adding one-step regretful action. Anderson et al. [27] proposes an agent formulated under Bayesian filtering with a global mapper. Hand-crafted decoding algorithms are also used as a post-processing technique but may lead to long trajectories and difficulty on joint optimization [2, 15, 28].

Navigation memory structures A recent emerging trend of navigation focuses on extending the agents with different types of memory structures. Simple structures: Parisotto et al. [11] propose a
tensor-based memory structure with operations that agent can use to access and perform navigation. Fang et al. [16] adopts transformer to extract historical memory information stored in a sequence. **Topological structures**: The landmark-based topological representation has shown to be effective in pre-exploration based navigation tasks without the need of externally provided camera poses or ego-motion information[12]. Laskin et al. proposes methods to sparsify the graphical memory through consistency checkings and graph cleanups [13]. Liu et al. uses a contrastive energy model for building more accurate edges in the memory graph [17].

**Graphical representation** Graph-based methods have been shown to be an effective intermediate representation for information exchange [29][30][31][32][33]. In image and video understanding, graphical representation is used for visual question answering [34][35], video captioning [36] or action recognition [37][38]. In robotics, Huang et al. [39] propose Neural Task Graph (NTG) as an intermediate modularized representation leading to better generalization. Graph Neural Networks are demonstrated to be effective in learning structured policies [26] and automatic robot design [40].

**Supervision strategies for imitation learning** The proper training of sequential models is challenging due to the drifting issues [41]. DAGGER [41] proposes to aggregate datasets with expert supervision provided for the sequences samples from student models. Scheduled sampling [42] tackles this problem through mixing the samples from both ground truth and student models. Professor forcing [43] shows a more effective approach through adversarial domain adaptation. OCD [44] adopts the online computed characters as ground truth for speech recognition. In this paper, we instead propose a graph-augmented strategy to provide expert supervisions, which alleviates the mismatch issue between instructions and new-computed expert trajectories [14].

### 3 Model

**Problem definition and setup** We follow the standard instruction-following navigation problem setting [11]. Given the demonstration dataset \( D = \{(x_i, \tau^*_i, ENV_i)\}_{i=1}^{2|D|} \), where \( x_i \) is the language instruction with length \(|x_i|\), \( \tau^* \) is the expert navigation trajectories \( \{a^{i}_1, a^{i}_2, ..., a^{i}_{|\tau^*_i|}\} \) and \( ENV_i \) is the environment paired with the data, the agent is trained to imitate the expert behaviours to correctly follow the instruction and navigate to the goal location. At each navigation step \( t \), the agent is located at the state \( s_t \), with observations \( o_t \) from the environment and performs an action \( a_t \in A_{s_t} \), where \( A_{s_t} \) is the decision space at state \( s_t \). In our task, decision space is the set of navigable locations [2].

To set up an agent, we build upon a basic navigation architecture in [14] and utilize the language encoder and the attention-based decoder. The agent encodes the language instruction \( x \) into a hidden encoding \( c = (c_1, c_2, ..., c_{|x|}) \) through an LSTM [45]. Conditioned on \( c \), the agent uses an attention-based decoder to model the distribution over the trajectory \( p(\tau|x, ENV) \). At every step, the decoder takes in the encoding \( c \), the observation \( o_t \) and a maintained hidden memory \( h_{t-1} \) to produce the per-step action probability distribution \( p(a_t|h_{t-1}, o_t, c) \). Note that such an navigation agent suffers from the constrained local action set and lacks the ability to perform long-range planning over the navigation space and to effectively correct errors along the navigation.

**Our approach** In this section, we introduce our Evolving Graphical Planner (EGP), an end-to-end global planner that navigates a agent through a new environment via re-defining the decision space and performing long-term plannings over graphs. With the graphical representation, the agent accumulates the knowledge about the unseen environment and has access to the entire action space. Each long-distance navigation is then reduced to one-step decision making, leading to easier and more efficient exploration and error correction. We also show that the graphical representation elicits a new supervision strategy for effective training of the imitation agent, which alleviates the mismatch issue in standard navigation agent training [14]. The global planning is performed on an efficient proxy representation, making the model scalable to navigation with longer horizon.

The proposed model consists of two core components: 1) an evolving graphical representation that generalizes the local action space; and 2) a planning core that performs multi-channel information propagation on a proxy representation. Starting from the initial information, the EGP agent gradually expands an internal representation of the explored space (Section 3.1) and perform planning efficiently over the graphs (Section 3.2).
Figure 2: Overall scheme of the Evolving Graphical Planner model. (1): Our graphical representation progressively expands during navigation (light-yellow = visited nodes; orange = current node; blue = potential action nodes; green = selected node). Based on the graphical representation, the agent performs planning over the actions, and selects the next action through student sampling. The top-K nodes on the current state will be kept in the graph. Note that the STOP action is also affiliated to the internal node as a leaf node, enabling the agent to travel back and stop at a previous visited location. (2): The model performs multi-channel planning on a proxy graph pooled from the entire graph representation. The refined node states are unpoled back to the entire graph and used to compute the probability distribution over actions. Best viewed in color.

3.1 Evolving Graphical Planner: Graphical representation

We start with introducing the notations of the representation. Let \( \mathcal{G}_t = \{ \mathcal{V}_t, \mathcal{E}_t, \mathcal{M}_t \} \) denote a graph at time \( t \), where \( \mathcal{V}_t = \{ v^1, v^2, \ldots, v^{\mid\mathcal{V}_t}\} \), \( v^i \in \mathbb{R}^d \) is the set of node embeddings, \( \mathcal{E}_t = \{ e^{ij} \} \), \( e^{ij} \in \mathbb{R}^d \) is the set of edge embeddings between node \( i \) and node \( j \), and \( \mathcal{M}_t \in \mathbb{R}^{\mid\mathcal{V}_t\times\mid\mathcal{V}_t\times\mid\mathcal{F}} \) is the graph connectivity tensor with function types. In our graph, nodes are separated into two types: the leaf nodes represent the possible actions (e.g. navigable locations) and internal nodes represent the visited locations, as shown in figure [2].

**Graph construction** The agent builds up the graphical representation progressively during navigation.Initialized as the empty set, the graph \( \mathcal{G}_t \) expands the node set, edge set and connectivity function tensor through the observations and local connection information given by the environment. When receiving an observation \( o_t \) and the set of information \( \{ a_{\alpha_t} \} \), over possible actions \( \{ \alpha_t \} \), at the new state \( s_t \), the agent maps the current location information \( o_t \) and action information \( a_{\alpha_t} \) through two separate neural networks, resulting to the node embedding \( v_{s_t} \) and \( \{ v_{a_{\alpha_t}} \} \). The incoming and outgoing edges of new nodes are determined through the local map information and the moving direction of the agent. There are three function types considered and stored in the tensor \( \mathcal{M}_t \): the forward and backward directions between two visited nodes, and the connection from the visited node to the leaf node. With the new graph \( \mathcal{G}_{t+1} \), the agent continues navigation and the loop repeats. To reduce the memory cost, the model has the option to selectively add nodes to the graph. We use top-K leaf nodes, ranked by confidence scores in policy distribution, to expand the graph, as shown in fig. [2].

**Navigation with graph jump** With graph \( \mathcal{G}_t \) representing the entire action space, the agent can easily choose to navigate to execute actions that have not been explored in previous visited locations. For the proposed action \( \alpha_t \) from the planner, the agent computes the shortest-path route based on the graph \( \mathcal{G}_t \) and plans the navigation route \( \tau' \). This allows the agent to “jump” around the full graph-defined action space and execute the unexplored actions through a single-step decision. The long-range decision space also makes error-correction easier for the agent: a single step decision is all the agent needs to backtrack to the correct location. With the navigation steps on the internal route \( \tau' \) generated through the graph, the agent keeps updating the hidden states \( h_t \) based on the observation \( o_t \) from the environments.

**Supervising imitation learning** The proper training of the imitation learning agent has been a challenging problem [44][41][43]. Student forcing with new computed route as supervision is a widely used solution in navigation [14][19][2]. However, the mismatch between the new route and language instructions could potentially lead to noisy and incorrect signals for learning language groundings and navigation. We provide a new graph-augmented solution for computing the supervision for each student sampled trajectory. Assume a metric \( \mathcal{M}(\cdot, \cdot) \) between two navigation trajectories (e.g. [46]). With a graph memorizing the entire possible action space, the subset of nodes in ground...
truth route $\tau^*$ is guaranteed to exist in $\mathcal{G}_t$. We choose the node $v$ on the $\tau^*$ that maximizes the metric $\mathcal{M}(\tau_i \cup \langle v \rangle, \tau^*)$ as the ground truth action for step $t$. This provides a “correction” signal that indicates the best action to take for correcting the mistake made by the agent. Since this supervision comes from ground truth trajectories, it is free of noisy signals that mismatches the instruction and does not require additional environment information to re-compute the new shortest-paths.

### 3.2 Evolving Graphical Planner: Planning Core

With the graphical representation $\mathcal{G}_t$, a straightforward method is to directly perform planning on the full graph using the embeddings of nodes and edges. However, a progressively growing graph can lead to high costs and limit the scalability of the model. Pre-exploration based methods often tackle this issue through performing offline sparsification with multiple rounds of cleanups on the pre-collected graphs containing full knowledge over the map, which are unsuitable under the online navigation settings. In this section, we show that, interestingly, the effective planning can be achieved not through the full graph, and present the second component of our model, where it performs a goal-driven information extraction dynamically on the entire graph $\mathcal{G}_t$ to build a condensed proxy representation $\mathcal{G}_t$ used for planning. Our model utilizes Graph Neural Networks (GNN) as a basic operator, which we explain at the end of the section.

**Proxy graph** Denote the proxy graph as $\bar{\mathcal{G}}_t = (\bar{V}_t, \bar{E}_t, \bar{M}_t)$, where $\bar{V}_t = \{ \bar{v}^1, \bar{v}^2, ..., \bar{v}^N \}$; $\bar{v}^i \in \mathbb{R}^d$, $\bar{E}_t = \{ e^{ij} \}$ and $\bar{M}_t$ are the pooled node embedding set, edge embedding set and connectivity matrix with function types respectively. $N$ is the size of proxy graphs and remains fixed for all steps. The proxy graph $\bar{\mathcal{G}}_t$ contains a fixed number of nodes invariant to the growing graph size in $\mathcal{G}_t$. We hypothesize that given the instruction information and the current states of the agent, there are only a subset of nodes providing useful information for planning. To construct the proxy representation, the model uses a normalized pooling similar to [47]. Differently, our graphical representation consists of a rich set of information including edge states, function types in connectivity matrices besides the node states. We describe the process of generating the proxy representation and corresponding planning as follows.

Given the entire graphical representation $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t, \mathcal{M}_t)$, the planner contains two functions, $\text{GNN}_L$ and $\text{GNN}_P$. The function $\text{GNN}_L$ takes in the agent state information and constructs the proxy representation through a lightweight neural network. Assume the pooling function uses $d_L$ as the graph dimension in the propagation model, and performs $K_L$ step message passing. We derive the pooling matrix as follows, using a small $d_L$:

$$A_t = \text{softmax} \left( \text{GNN}_L(\mathcal{G}_t, [h_t; c]; K_L, d_L) \right) \in \mathbb{R}^{|V_t| \times |V_t'|}$$

The normalized pooling matrix is the attention weights on the entire graph and extracts relevant information conditioned on the agent states $h_t$ and instruction encodings $c$ for further planning. To help the description, we denote the concatenated matrix of all node vectors as $V_t \in \mathbb{R}^{|V_t| \times d}$, the concatenation of all edge vectors as tensor $E_t \in \mathbb{R}^{|V_t| \times |V_t'| \times d}$. The concatenation order is aligned with the connectivity function matrix. The following operations are used for deriving the proxy representation.

$$V_t = A_t^T V_t \in \mathbb{R}^{|V_t'| \times d}$$
$$\bar{M}_{t,i} = A_t^T M_{t,i} A_t \in \mathbb{R}^{|V_t'| \times |V_t'|}, i \in \{1, 2, ..., |F| \}$$
$$\bar{E}_{t,i} = A_t^T E_{t,i} A_t \in \mathbb{R}^{|V_t'| \times |V_t'|}, i \in \{1, 2, ..., d \}$$

where $\bar{M}_{t,i}$ is a non-negative matrix indicating the weights of connectivity among nodes for function type $i$, $\bar{E}_{t,i}$ is the matrix at $h_{i,t}$ dimension for the edge state tensor. The pooled tensors $V_t$ and $E_t$ are corresponded to the node set $\bar{V}_t$ and edge set $\bar{E}_t$ of the proxy graph $\bar{\mathcal{G}}_t$.

**Planning** The planning of the navigation agent is achieved through propagating information among nodes in the proxy graph, conditioned on the agent state information $h_t$ and instruction encodings $c$. Denote the graph dimension used in the propagation model as $d_P$, the number of steps for planning.

---

1 The planning term here refers to an implicit process using message passings to maximize a fitness metric (such as nDTW [45] imposed through $\mathcal{M}(\cdot, \cdot)$), compared to other methods that explicitly maximize a fitness score to perform planning.
message passing operations as $K_P$, the refined node embedding of the proxy graph is derived through GNN$_P$, $(V_1, [h_i; c]; K_P, d_P)$, with $d_P$ controlling the capacity of the function. The refined node embedding contains the information involving the neighboring nodes (visited locations and unexplored actions), the state of agent, the instruction, and the connectivity types between nodes. With the globally refinement step, node embedding vector is unpooled back to the original graph through $V^* = \bar{V}_t \cdot A^T_i$. With the update node representation containing both the current state of the agent and the full action-observation space in the history, the distribution over actions is generated based on the node vectors:

$$\hat{a}_i^t = f_{out}(\bar{h}_t, v_i^t); p(a_i^t|\bar{h}_t, V_t) = \exp(\hat{a}_i^t)/\sum_j \exp(\hat{a}_j^t)$$  \hspace{1cm} (1)

where the $f_{out}$ function is a dot-product with linear mapping $(W_{hv} \cdot \bar{h}_t)^T v_i^t$ parameterized by $W_{hv}$.

**Multi-channel planning** The model is further strengthened with the ability to perform multi-channel planning over the graph $G_t$. Instead of only using one proxy graph representation $G_t$ to extract information and perform propagation, we find it useful to learn a set of proxy graphs $\{G_t\}$ and perform planning independently on each of them. The final information are aggregated through summation over the embedding across the channels. The final policy over actions is generated through the same operation described in Eqn. 1.

**Training objective** We train our full agent through the standard maximum likelihood objective using cross-entropy loss. Given the demonstration $D$, the loss function to optimize is formulated as:

$$\mathcal{L} = \mathbb{E}_{(\tau^*, x, \text{ENV}) \sim D} \left[ \sum_t -y^t_i \log(p(a_i^t|\bar{h}_t, c = \text{LSTM}(x))) \right]$$  \hspace{1cm} (2)

where $\tau^*$ is the ground truth trajectory, $y^t_i$ is the ground truth label on action $i$ at step $t$, generated through the graph-augmented supervision using the information of $\tau^*$, as described in Sec. 3.1.

### 3.2.1 Message Passing Operations

In this subsection we explain the Graph Neural Network (GNN) operator used in the EGP. As the operator is used in both pooling and planning, we describe it as a general function which takes in a graph $G$ and a general context information $r$ (e.g., the agent hidden state and language encodings), with hyper-parameters $K$ and $d_G$. Formally, given the input graph $G = (V_0, E_0, M)$ and the context vector $r$, the function $\text{GNN}(G, r; K, d_G)$ generates the refined node vectors $V_{K+1}$ after $K$ steps of message passings, where $d_G$ is the vector dimension in the propagation model. The subscription on node and edges denotes the index of message passing iterations, with a slight abuse of notation. The $\text{GNN}(\cdot, \cdot; K, d_G)$ function contains two components: an input network and a propagation network.

**Input network** Along with the initial vectors for nodes and edges, the input network considers the context vector $r$ as a shared additional information across nodes and edges. The input network maps the context vector with node and edge vectors respectively into two fixed-size embedding as follows:

$$v^i_1 = f_n(v^0_i, r); e^{ij}_1 = f_n(e^{ij}_0, r); v_1^i, e_1^{ij} \in \mathbb{R}^{d_G}$$  \hspace{1cm} (3)

where $f_n$ is a neural network, and the generated embedding vectors are used for message communications among graph nodes in the propagation model.

**Propagation network** The propagation model, taking in the mapped embedding vectors $\{v^i_1\}$, consecutively generates a sequence of node embedding vectors $v^i_k, k \in \{1, ..., K+1\}$ for each node. At step $k$, the propagation operation updates every node through computing and aggregating information from the neighborhood nodes. The process is executed in the order:

$$e^{ij}_k = \sum_n M_{i,j,n} \cdot f_n(v^i_k, v^j_k, e^{ij}_k) \in \mathbb{R}^{d_G};$$  \hspace{1cm} (4)

$$v^i_{k+1} = g \left( \sum_{j \in \mathcal{N}_i} e^{ij}_{k}, v^i_k \right) \in \mathbb{R}^{d_G}$$  \hspace{1cm} (5)

where $f_n(\cdot, \cdot; \cdot) : \mathbb{R}^{3d} \rightarrow \mathbb{R}^{d}$ is the message function. Function $g(\cdot) : \mathbb{R}^{2d} \rightarrow \mathbb{R}^{d}$ is the aggregator function that collects messages back to node vectors. $\mathcal{N}_i$ represents the neighbours of node $i$ in the graph. The refined node vector set $V_{K+1} = \{v^i_{K+1}\}$, containing global information from the whole graph is mapped through a matrix $W_{out}$ to recover the input node dimension $d_0$ and is used as the output of the $\text{GNN}(\cdot, \cdot)$ function for either pooling or planning component, as described in Sec. 3.2.
4 Experiments

4.1 Experimental setup

Datasets We evaluate our method on the standard benchmark datasets for Vision-and-Language Navigation (VLN). The VLN task is built upon photo-realistic simulated environments with human-generated instructions describing the landmarks and directions for navigation routes. Starting at a random sampled location in the environment, the agent needs to follow the instruction to navigate through the environment. There are two datasets commonly used for VLN: (1) Room-to-Room (R2R) benchmark with 7,189 paths, each associated with 3 sentences, resulting in 21,567 total human instructions. The paths are produced through computing shortest paths from start to end points; (2) Room-for-Room (R4R), which extends the R2R dataset by re-emphasizing on the necessity of following instructions compared to the goal-driven definition in R2R. The R4R dataset contains 278,766 instructions associated with twisted routes connecting two shortest-path trajectories in R2R. The dataset details are summarized in Table 1.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Train</th>
<th>Val:seen</th>
<th>Val:unseen</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>R2R</td>
<td>14,039</td>
<td>1,021</td>
<td>2,349</td>
<td>4,173</td>
</tr>
<tr>
<td>R4R</td>
<td>233,532</td>
<td>1,035</td>
<td>45,234</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 1: Dataset statistics.

Implementation details We follow and adopt the co-grounding agent (w/o auxiliary loss) as our base agent. As the standard protocol for R2R, visual features for each location are pre-computed ResNet features from the panoramic images. In the Evolving Graphical Planner, we use 256 dimensions as the graph embedding size for both the full graph and the proxy graph. The propagation model uses three iterations of message passing operations. For every expansion step, the default setting adds all the possible navigable locations into the graph (top-K is set to 16, the maximum number of navigable location in both datasets). For student-forced training, we use graph-augmented ground truth supervision throughout the experiments except for the ablation study on supervision methods. The model is trained jointly, using Adam with 1e-4 as the default learning rate.

4.2 Room-to-Room benchmark

Evaluation metrics We follow prior works on the R2R dataset and report: navigation error (NE) in meters, lower is better; Success Rate (SR), i.e., the percentage of navigation end-locations which are within 3m of the true global location; Success Rate divided by path Length in meters (SPL); and Oracle Success Rate (OSR), i.e., whether the path at any point passes within 3m of the goal state.

4.2.1 Comparison with prior art

Architectures for comparison We compare our model with the following state-of-the-art navigation architectures: (1) the Seq2Seq agent that translates instructions to actions; (2) Speaker-Follower (SF) agent that augments the dataset with a speaker model; (3) the Reinforced Cross-Modal (RCM) agent using modal-alignment score as intrinsic reward for reinforcement learning; (4) the Self-Monitoring (Monitor) agent that uses a co-grounding module and a progress estimation component to increase the progress alignment between texts and trajectories; (5) the Regretful agent that uses the Regretful module and Progress Marker to perform one-step rollback; (6) the Ghost with Bayesian filters.

Results We report results in Table 2. We train our models both by using only the standard demonstration and by augmenting the dataset with the synthetic data containing 178,330 instruction-route pairs generated by the Speaker model. On the Val Unseen split, we observe that just through using the EGP module (without synthetic data augmentation), the performance of agent can be increased over the baseline agent by 0.86 meters on NE (from 6.20 to 5.34), by 9% on SR (from 43% to 52%), by 5% on SPL (from 36% to 41%), and by 13% on OSR (from 52% to 65%). Our path length remains short, at 13.7 meters compared to 12.8m for baseline, 14.8m for RCM and 15.2m SF (not shown in the table). Most notably, our EGP agent with synthetic data augmentation outperforms prior art on all metrics, across both the validation-unseen and the test set. Concretely, on the test set we achieve a 0.35 meters reduction in NE (from 5.69 to 5.34), a 5% improvement on SR (from 48% to 53%), a 2% improvement on SPL (from 40% to 42%), and a 2% improvement on OSR (from 50% to 61%) over the best performing prior model on each metric respectively.
Table 2: We compare our architecture with previous state-of-the-art architectures on the Val Unseen and Test splits of R2R [1]. (∗: models using additional synthetic data. %: numbers in percentage). We also compare our results with a graph-search based post-processing method [15] which uses longer path-length, hand-crafted rules and other information such as progress monitor and speaker scores.

Discussion of other works Note that there are other works contributing to this benchmark through non-architecture approaches: using more data (6,582K) for BERT-type pre-training [4]; exploiting web data [51]; adding extra tasks [52]; adding dropout regularization [53]; different settings of evaluation (fusing information from three instructions) [54, 55]; post-processing decoding method [15]. We contribute a clean and differentiable backbone navigation architecture and these works can potentially be useful as complementary approaches.

4.2.2 Ablation studies

We now justify the design choices of our model by analyzing the individual components. In addition to the metrics above we also include Path Length (PL) for completeness. Results are summarized in table 3, with the last row depicting our model with the default settings.

Global planning We verify the importance of global planning through controlling the top-K expansion rate for the graphical representation. In R2R dataset, there are maximally 16 navigable locations for each state. With a smaller expansion rate, the EGP planner has less expressive power on exploiting global information from the environments. As seen in the top group of rows of table 3, with smaller top-K, the path becomes shorter (fewer options to explore) and the accuracy of the model consistently drops, indicating the importance of global planning.

Planner implementation Next we analyze the effects of message passing (mp) steps and the number of channels used on our planner module. The results are summarized in the second group of rows in table 3. Through the information propagation operations, our model achieves a 8% increase on SR (from 42% with mp=0,channel=1 to 50% with mp=3,channel=1). With more independent planning channels, we can obtain a further 2% improvement on SR (from 50% with mp=3,channel=1 to 52% for our default setting with mp=3,channel=3, last row). To verify whether the increase is due to more parameters in the model, we also add a comparison through using a single channel planner with three times larger graph dimensions (768), which shows no similar effect to the multi-channel models.

Supervision methods Finally, we compare our method a navigation agent trained by the standard student forcing. The standard supervision used for student forcing requires recomputing the new shortest-path route to the goal from each location, leading to potential noisy data and larger generalization error, shown in the second-from-last row of table 3.

4.3 Room-for-Room benchmark

Evaluation metrics The Room-for-Room dataset emphasizes on the ability of correctly following instructions instead of solely on reaching the goal locations. We follow the metrics in [18,46] and
mainly compare our results on Coverage weighted by Length Score (CLS) that measures the fidelity of the agent’s path to the reference, weighted by the length score, and the Success rate weighted normalized Dynamic Time Warping (SDTW) and normalized Dynamic Time Warping (nDTW) that measure the spatio-temporal similarity of the paths by the agent and the expert reference.

Architectures for comparison We compare our model with the Speaker-Follower model [2], Reinforced Cross-Modal agent [50] trained under goal-directed and fidelity-oriented rewards, reported in [18], and the Perceive Transform Act (PTA) agent [56] using more complex multi-modal attentions.

Results analysis We summarize the results in table 4. Note that all previous state-of-the-art methods require the mixed training between imitation learning and reinforcement learning objectives. The student forcing method leads to goal-oriented supervisions and harms the ability of following instructions for agents [18]. Our model is the first that successfully train the navigation agent through pure imitation learning on the R4R benchmark, due to the benefit of the graphical representation, the powerful planning module and the new supervision method. We obtain a consistent margin across all metrics. Specifically, our model outperforms other architectures by 7.0, 5.0 and 4.9 on the fidelity-oriented measurements CLS, nDTW and SDTW respectively. Also, although using a global search mechanism, our model maintains a relatively short path length, which is difficult in other rule-based global search algorithms [2, 15].

<table>
<thead>
<tr>
<th>Models</th>
<th>Type</th>
<th>PL</th>
<th>NE ↓</th>
<th>SR ↑</th>
<th>CLS ↑</th>
<th>nDTW ↑</th>
<th>SDTW ↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>IL+RL</td>
<td>23.6</td>
<td>10.4</td>
<td>13.8</td>
<td>22.3</td>
<td>18.5</td>
<td>4.1</td>
</tr>
<tr>
<td>Speaker-Follower [13]</td>
<td>IL+RL</td>
<td>19.9</td>
<td>8.47</td>
<td>23.8</td>
<td>29.6</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RCM + goal-oriented [18]</td>
<td>IL+RL</td>
<td>32.5</td>
<td>8.45</td>
<td>28.6</td>
<td>20.4</td>
<td>26.9*</td>
<td>11.4*</td>
</tr>
<tr>
<td>RCM + fidelity-oriented [18]</td>
<td>IL+RL</td>
<td>28.5</td>
<td>8.08</td>
<td>26.1</td>
<td>34.6</td>
<td>30.4*</td>
<td>12.6*</td>
</tr>
<tr>
<td>PTA low-level [56]</td>
<td>IL+RL</td>
<td>10.2</td>
<td>8.19</td>
<td>27.0</td>
<td>35.0</td>
<td>20.0</td>
<td>8.0</td>
</tr>
<tr>
<td>PTA high-level [56]</td>
<td>IL+RL</td>
<td>17.7</td>
<td>8.25</td>
<td>24.0</td>
<td>37.0</td>
<td>32.0</td>
<td>10.0</td>
</tr>
<tr>
<td>EGP (ours)</td>
<td>IL</td>
<td>18.3</td>
<td>8.0</td>
<td>30.2</td>
<td>44.4</td>
<td>37.4</td>
<td>17.5</td>
</tr>
</tbody>
</table>

Table 4: Comparison across methods on R4R Val Unseen split. Path Length (PL) is reported as a reference. (*Note that we refer to the numbers from [46] as [18] did not report DTW-based results)

5 Conclusion

In this work, we proposed a solution to the long-standing problem of contextual global planning for vision-and-language navigation. Our system based on the new Evolving Graphical Planner (EGP) module outperforms prior backbone navigation architectures on multiple metrics across two benchmarks. Specifically, we show that building a policy over the global action space is critical to decision making, the graphical representation can further elicit a new supervising strategy for student forcing in navigation, and, interestingly, the actual planning can be achieved through a proxy graph rather than the actual topological representation which leads to high costs in both time and memory.
Broader Impact

This work has several downstream applications in areas like autonomous navigation and robotic control, especially through the use of natural language instruction. Potential downstream uses of such agents range from healthcare delivery to elderly home assistance to disaster relief efforts. We believe that imbuing these agents with a global awareness of the environment and long-term planning will enable them to handle more challenging tasks and recover gracefully from mistakes. While the graph-based approach we propose is scalable and easy to manipulate in real time, future research can address computation challenges and increase the planning time-scale to enable better decision making.

Acknowledgments and Disclosure of Funding

This work is partially supported by King Abdullah University of Science and Technology (KAUST) Office of Sponsored Research (OSR) under Award No. OSRCRG2017-3405 and by Princeton University’s Center for Statistics and Machine Learning (CSML) DataX fund. We would also like to thank Felix Yu, Angelina Wang and Zeyu Wang for offering insightful discussions and comments on the paper.

References


