Rethinking the Spatial Route Prior in Vision-and-Language Navigation

Xinzhe Zhou[1], Wei Liu[2], Yadong Mu[1*]
[1]Peking University, [2]Tencent Corp.

Abstract

Vision-and-language navigation (VLN) is a trending topic which aims to navigate an intelligent agent to an expected position through natural language instructions. This work addresses the task of VLN from a previously-ignored aspect, namely the spatial route prior of the navigation scenes. A critically enabling innovation of this work is explicitly considering the spatial route prior under several different VLN settings. In a most information-rich case of knowing environment maps and admitting shortest-path prior, we observe that given an origin-destination node pair, the internal route can be uniquely determined. Thus, VLN can be effectively formulated as an ordinary classification problem over all possible destination nodes in the scenes. Furthermore, we relax it to other more general VLN settings, proposing a sequential-decision variant (by abandoning the shortest-path route prior) and an explore-and-exploit scheme (for addressing the case of not knowing the environment maps) that curates a compact and informative sub-graph to exploit. As reported by [33], the performance of VLN methods has been stuck at a plateau in past two years. Even with increased model complexity, the state-of-the-art success rate on R2R validation-unseen set has stayed around 62% (for single-model) and 73% (with beam-search and ensemble). We have conducted comprehensive evaluations on both R2R and R4R, and surprisingly find that utilizing the spatial route priors may be the key of breaking above-mentioned performance ceiling. For example, on R2R validation-unseen set, when the number of discrete nodes explored is about 40, our single-model success rate reaches 73%, and increases to 78% if a Speaker model is ensembled, which significantly outstrips previous state-of-the-art VLN-BERT with ensemble and clearly recalibrates the top records in VLN.

1. Introduction

Humans have a long-standing dream of creating intelligent robots that can accomplish different tasks following our instructions. Recent research about autonomous navigation has made tremendous advances towards this goal. In particular, the work of Anderson *et.al.* [3] attracted much attention as it involved both vision-language interaction and embodied navigation, known as Vision-and-Language Navigation (VLN). This task requires an autonomous agent to reach an expected goal position guided by purely natural language instructions. The release of several VLN benchmarks, including the Room-to-Room (R2R) [3], Room-for-Room [15] (R4R) and TOUCHDOWN [4], has spurred the rapid development of VLN models. The key performance metrics, such as success rate (SR), have witnessed continual improvement in the past years. Interestingly, a majority of existing works are formulated under the reinforcement learning (RL) paradigm. We are surprised to notice that most of previous works have ignored the spatial route prior inherent in the VLN task. For example, it is explicitly explained [3] that all ground-truth routes in R2R are direct-to-goal shortest-paths. Likewise, the routes in TOUCHDOWN [4] are obtained via optimized route-calculating APIs in Google Map.

This work aims to harness various types of spatial route priors for enhancing the performance of VLN models. Our formulation starts by considering a naive case with shortest-path route prior and all navigation map known. As a direct derivation of the shortest-path prior, a pair of origin-destination would often uniquely determine the internal path that an agent goes through (the existence of multiple shortest paths for the same node pair is possible yet very rare). Therefore, under this strictly-constrained VLN setting, finding the best-matching route for an instruction boils down to choosing a node in the navigation map as the destination. We can thus frame the path-finding problem as an ordinary classification task over all valid nodes, which is more amenable to numerical optimization and tends to bring superior performances in comparison with conventional RL-based models as empirically verified in our experiments. The merits of the classification-based formulation are two-fold. First, RL-based methods suffer from an exponential number of candidate paths, while our proposed formulation reduces it to be linear to the number of nodes in the navigation scenes. For example, the R2R data has an average vertex degree of 4.1 and the ground-truth path contains 4 to 6 steps. Simple calculations reveal

that RL-based methods would consider about 283 to 4,750 possible candidates, compared with 117 on average in our classification-based formulation. A second merit is classification on the whole-path level could make global inference about the route. RL-based agents instead generate the route from sequential local decisions. The issue of locality is only partly mitigated in recent works that integrate backtracking [16, 21] or looking-ahead [27] mechanisms.

In realistic applications, the topology of navigation scenes is often unknown, and the shortest-path prior may not hold on a few challenging VLN benchmarks, such as R4R [15]. We thus further propose two variants of the basic classification-based VLN model. First, the shortest-path prior is relaxed to be more general, allowing a ground-truth path to be the composition of an arbitrary number of local shortest sub-paths. Based on this relaxation, we adapt the classification model into a progress-aware sequential-decision variant, which resembles previous RL agents but classifies among the shortest sub-paths instead of all neighboring edges. Secondly, the assumption of knowing the topology of navigation maps is removed. We develop an informative partial topology (sub-graph) of the environment on-the-fly by re-purposing a weak Follower [8] model, upon which the (sequential) classification could be conducted then.

Our experiments on R2R and R4R all clearly show that by exploring only sub-graph of moderate size, our model could outperform previous state-of-the-art methods with significant margins. For example, on the R2R validation-unseen set, when the number of discrete nodes explored is about 40, our single-model success rate reaches 73%, and increases to 78% if a Speaker [8] model is ensembled, which even out-performs the 3-model ensembled VLN-BERT [22] with similar exploration. Our evaluations have provided strong evidence that the spatial route priors may be the key for breaking the performance ceilings as reported by recent VLN works [33].

2. Related Work

In an early VLN work [3], Anderson *et.al.* designed a simple sequence-to-sequence (Seq2Seq) agent that operated directly in low-level action spaces and was trained by purely Imitation-Learning (IL). This model set a baseline success-rate (SR) that was much lower than human performance (20% v.s. 86%). After it, numerous algorithms were continually developed for improvement in different aspects. For instance, [30] changed the pure-IL training to a hybrid of model-based and model-free Reinforcement-Learning (RL). [29] further designed an ensemble of IL and RL training paradigm, which has now become the default choice. The work of [8] revised the original low-level action space to a more high-level panoramic space, which greatly reduced the sequence length and was followed by almost all later works. Besides, [8] also proposed a dual Speaker-Follower framework, which helped in both data augmentation and inference-time ensemble. Some later works attempted to introduce heuristic mechanism into the agent like backtracking / regret [16, 21], active exploration [27]. Another line of research was trying to improve the unseen generalization through either pre-training [13, 10, 22], auxiliary supervision [20, 31, 32, 28], or training-data processing [25, 9, 23]. The basic Seq2Seq structure had also been improved by introducing cross-modal attention [29] and fine-grained relationship [12], utilizing the semantic or syntactic information of languages [24, 18], reformulating under a Bayesian framework [2], and combining long-range memory for global decision [6, 26]. Besides R2R, some later works also proposed more challenging datasets like Room-for-Room (R4R) [15] and TOUCHDOWN [4].

3. Our Approach

3.1. The Naive Case: Shortest-Path Prior + Known Map

Assume that a navigation environment can be represented by an edge-weighted graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V}$ collects all valid observation positions and $\mathcal{E}$ covers all direct routes within two neighboring nodes. In a typical setting of VLN, an agent is initially located at some $n_s \in \mathcal{V}$ and moves once receiving an l-word navigation instruction $\mathcal{I} = (w_1, w_2, \ldots, w_l)$. For the instruction $\mathcal{I}$, there always exists a ground-truth path $\mathcal{P}_{n_s,gt_t} = (n_{gt_1}, n_{gt_2}, \ldots, n_{gt_t})$, where $n_{gt_1} = n_s$ and $(n_{gt_j}, n_{gt_{j+1}}) \in \mathcal{E}$ for any step j. t is the length of the path and often varies over different instructions.

Let us first consider a naive case: the full information of graph $\mathcal{G}$ is known and each ground-truth path is the shortest between the origin-destination pair, namely the agent operates with a shortest-path prior and known map. Such a setting paves a way to learn an effective albeit simple VLN model which is essentially different from popular RL-based solutions. In specific, given an instruction $\mathcal{I}$ and a starting node n_s, the shortest path $\mathcal{P}_{n_s,n_i}$ between n_s and n_i can be uniquely determined via standard Floyd or Dijkstra algorithm [7]. This fact critically leads to classification-based path-finding solver conditioned on $\mathcal{I}$ and n_s:

$$\arg\max_{n_i} \mathcal{F}(\mathcal{P}_{n_s,n_i}, \mathcal{I}), \ \ \forall\, n_i \in \mathcal{V}, \tag{1}$$

where $\mathcal{F}(\cdot)$ stands for any well-defined path-instruction scoring function.

One of the ambitions in this paper is to demonstrate that the simple classification in Eqn. 1, compared with various sophisticated RL-based policies in existing work, can serve as a major innovation that significantly advances the state-of-the-art of VLN. To this end, we directly adopt a

Algorithm 1 Decompose a route into sequential shortest sub-paths

Input: A route $\mathcal{P} = (n_1, n_2, \ldots, n_t)$ and the corresponding graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$;
Output: The decomposed shortest-paths $\mathcal{P}'$;
Compute the shortest-paths between any two nodes in $\mathcal{G}$; Initialize $\mathcal{P}'$ to an empty list and *i* to 1;
while $i \leq t$ **do**
  $j \leftarrow i$;
  while $j \leq t$ and $(n_i, n_{i+1}, \ldots, n_j)$ is the shortest-path between n_i and n_j **do**
    $j \leftarrow j + 1$;
  end while
  Append $(n_i, n_{i+1}, \ldots, n_{j-1})$ to the tail of $\mathcal{P}'$ and mark (n_i, n_{j-1}) as a local starting-ending node pair;
  $i \leftarrow j - 1$;
end while

naive off-the-shelf dual-LSTM [11] based model to implement $\mathcal{F}$, which has been widely adopted in previous practice [8, 20, 25, 32, 27]. In detail, two separate LSTMs are used to encode the path and the instruction, denoted as $\text{LSTM}_{\text{path}}$, $\text{LSTM}_{\text{instr}}$ respectively. Their final hidden-state vectors are concatenated and fed into an MLP, obtaining a compatibility-indicating score. The optimization of this model can be guided by an ordinary cross-entropy loss calculated on exhaustively-sampled path-instruction pairs on the training navigation scenes, where the ground-truth pairs are known.

One intriguing property of the VLN task is the judgment of a successful navigation. A commonly-adopted metric, success rate (SR) [3], regards any navigational path that ends at any position within specific distance (a user-settable hyper-parameter, say 3 meters) of the ground-truth goal to be correct. This subtly diverges from standard classification problems with one-hot ground-truth labels. To fully harness such property for enhancing the navigation success rate, we stick to the ordinary Softmax neural layer for generating activation scores, yet at the inference time, a *neighboring score aggression* scheme is proposed to compensate those sufficiently close yet not exact solutions. Namely, the optimal destination is calculated via:

$$\arg\max_{n_i} \ \text{sum}_{n_j = n_i \text{ or } n_j \in \mathcal{N}(n_i)} \{\mathcal{F}(\mathcal{P}_{n_s, n_j}, \mathcal{I})\}, \ \forall\, n_i \in \mathcal{V}, \tag{2}$$

where the index set $\mathcal{N}(n_i)$ contains all nodes within a small distance (3 meters in our experiments) of the node n_i, and $\text{sum}_{n_j \in \mathcal{N}(n_i)}$ defines an operator that sums all scores from $\mathcal{N}(n_i)$. More details about hyperparameters and training schedules are summarized in the supplementary material.

3.2. Relaxation to General Route Prior + Known Map

As stated before, the shortest-path prior does not always hold on all VLN benchmarks, such as the newly-constructed R4R [15] dataset. To extend the idea in Section 3.1 to more general cases, this section elaborates on a second scheme that addresses a setting with more general spatial route priors: each path is decomposed into several non-overlapping sub-paths, each of which admits a local shortest path with respect to its starting and ending nodes. Our proposed method starts from decomposing an arbitrary ground-truth route into consecutive shortest sub-paths. Note that the desired decomposition may not be unique and multiple valid ones may exist. In the VLN task, a decomposition with minimal number of sub-paths is favored, since it will require fewer route-finding operations. To this end, we propose a one-pass process that greedily pursues a local shortest sub-path with maximal steps. The pseudo-code for the process is found in Algorithm 1.

Given an instruction $\mathcal{I}$ and the corresponding ground-truth path $\mathcal{P}$, note that the output of Algorithm 1 is a list of starting-ending node pairs. Denote them as $(n_s^{(1)}, n_e^{(1)}), (n_s^{(2)}, n_e^{(2)}), \ldots, (n_s^{(K)}, n_e^{(K)})$, where K counts the sub-paths return by Algorithm 1. This inspires us to tailor the method described in Section 3.1 to a sequential-decision variant. Concretely, the agent attempts to accomplish a route-finding task in K rounds. At round k, the model is optimized to navigate the agent from $n_s^{(k)}$ to $n_e^{(k)}$, ideally forming a partial path $n_s^{(1)} \rightarrow n_e^{(k)}$ with k shortest sub-paths. We formulate each round as a classification problem as in Section 3.1, with $n_s^{(k)}$, $n_e^{(k)}$ to be the starting node and locally ground-truth destination node respectively. Such iterative decisions progressively expand the partial path, targeting the ground-truth path $\mathcal{P}$. As the total number of rounds K is unknown for the agent, we introduce a special learnable candidate to indicate the "Stop" action, which is jointly optimized through the classification process in each round. An example from R4R illustrating the above process is shown in Figure 1.

The function $\mathcal{F}(\cdot)$ in Eqn. (1) gauges the path-instruction compatibility. However, at each round of above sequential-decision process, $\mathcal{F}(\cdot)$ will be fed with a full instruction yet partial path, leading to imprecise scores. To remedy this, we propose a progress-aware scoring function. At specific round, suppose the current pursued path is $(n_1, n_2, \ldots, n_{\tilde{t}})$, with $\tilde{t}$ nodes. The dual-LSTM outputs:

$$h_1^I, h_2^I, \ldots, h_l^I = \text{LSTM}_{\text{instr}}(w_1, w_2, \ldots, w_l), \tag{3}$$

$$h_1^P, h_2^P, \ldots, h_{\tilde{t}}^P = \text{LSTM}_{\text{path}}(n_1, n_2, \ldots, n_{\tilde{t}}), \tag{4}$$

where h_j^I and h_j^P represent the j-th step hidden states of the encoded instruction and path, respectively.

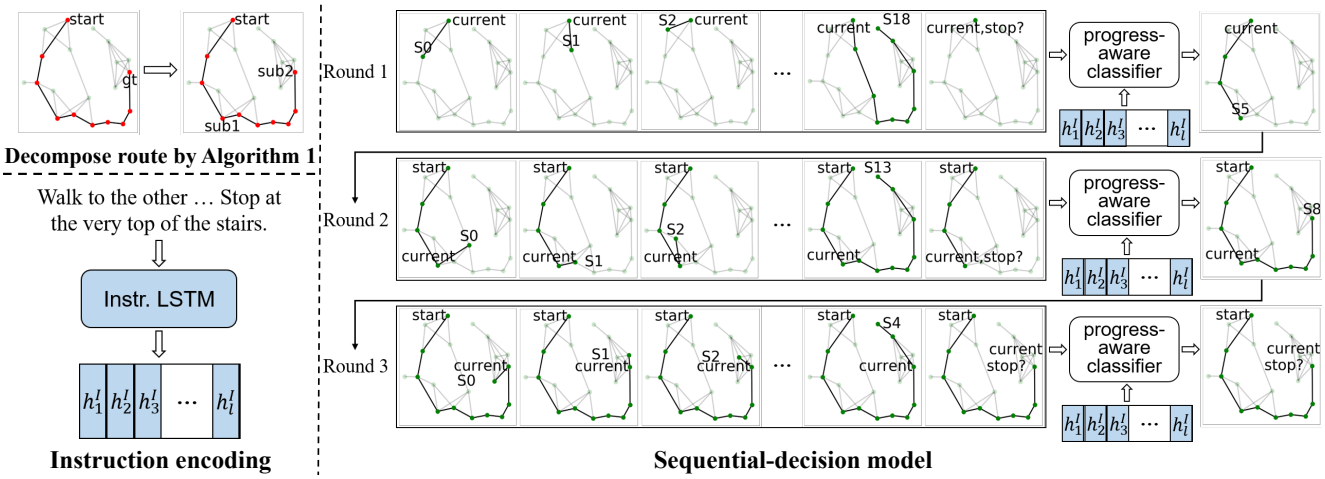

Figure 1: An example illustrating the route decomposition and the sequential-decision process.

The main fact that we harness here is h_t^P should accurately match some one in $\{h_1^I, h_2^I, \ldots, h_l^I\}$ if the current round successfully finds an optimal partial path. Inspired by soft-DTW [5], we fetch the hidden state at every step of the instruction and concatenate each of them with the path final state to compute the scores corresponding to different progresses / steps using an MLP, according to

$$s_j = \text{MLP}([h_j^I, h_t^P]), \ \forall\, j = 1, 2, \ldots, l. \quad (5)$$

Then, we conduct an aggregation to obtain a final score s:

$$\begin{aligned} s &= \text{Aggr}_{\text{soft}}\left(s_1, s_2, \ldots, s_l\right) \\ &= \log\left(\exp(\frac{s_1}{\gamma}) + \exp(\frac{s_2}{\gamma}) + \cdots + \exp(\frac{s_l}{\gamma})\right). \quad (6) \end{aligned}$$

Simply put, $\text{Aggr}_{\text{soft}}$ is a soft maximum operation which behaves like soft-attention [19] to aggregate different progresses. γ is set to 1 by default. The comparison of $\text{Aggr}_{\text{soft}}$ and the original MLP function is shown in Section 4.3.

The above model is akin to existing RL-based agents [8, 25] in that both operate sequentially and form the whole route from sub-paths. Nonetheless, there are several important differences between them. First, each decision in previous RL-based agents chooses one more edge in the route, leading to many steps to return the ground-truth route. In contrast, each decision in our model will include a longer sub-path, thus greatly reducing the number of required decisions. Take R4R as an example, with previous agents, the average length of the decision sequence is about 12, while with our decomposition as Algorithm 1, the average number of the sub-paths is only 3.3. Besides, our sequential model inherits the global-view merit of the classification model in Section 3.1. Compared with previous methods that make local decisions in the instant neighbors, our model classifies among the shortest-paths from current node which extends the view to several steps away and enables a more global decision. This helps alleviate the chance that the model falls into local minimum (w.r.t. the loss).

The optimization of the above model follows the previous practice [29], by jointly using Imitation learning (IL) and RL. For IL, we use teacher-forcing and feed the teacher choice for each decision by Algorithm 1. For RL, we use the CLS-based reward proposed in [15], as it encourages to reach not only the goal but the fidelity to the ground-truth route. More details about the training are included in the supplementary material.

3.3. Relaxation to Unknown Map

Both of above models operate well when the environment topology is known beforehand, such that the shortest-paths could be exactly computed. It is more realistic to assume the environment is totally unseen before the navigation. To generalize our model to the settings without knowing the navigation map, we here develop an explore-and-exploit scheme. It works as follows: first, we run an agent with an exploration policy to construct an informative sub-graph of the environment. The policy can be any off-the-shelf one reported in existing VLN works, and we choose a simple Follower [8] model with the state-factored search (a.k.a. beam-search). Next, a classification model or a sequential-decision model, as described in previous sections, is chosen according to the route prior. The model runs on the sub-graph found in the first step, returning the path with highest path-instruction score.

Our explore-and-exploit scheme is similar to the popular beam-search and re-ranking paradigm [8, 22], in the sense that both need to explore the environments before the final decision. But the difference is that we construct a sub-graph from all the explored nodes and exploit the (relaxed) prior

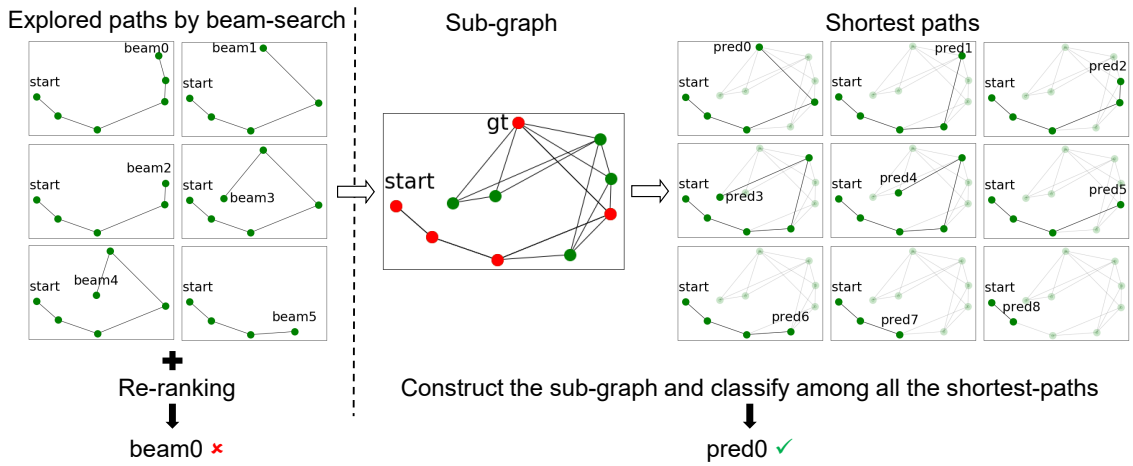

Figure 2: An example illustrating our explore-and-exploit scheme (right) with comparison to previous beam-search and re-ranking paradigm (left) from R2R.

over the sub-graph. While for re-ranking methods, they collect the candidates (beams) and only choose one best from them, ignoring the graph topology and route priors. An example illustrating the difference is shown in Figure 2.

4. Experiment

4.1. Results on R2R Dataset

Dataset. Room-to-Room (R2R) [3] is among the first and most widely adopted datasets for benchmarking VLN performance. It contains rich panoramic scenes of 90 different houses / environments with manually annotated instruction-route pairs. The dataset is split into `train` (61 environments, 14,025 instructions), `val-seen` (56 environments, 1,020 instructions), `val-unseen` (11 environments, 2,349 instructions), and `test-unseen` (18 environments, 4,173) sets.

Evaluation Metric. Following [3, 1], we use five metrics: (1) *Success Rate* (SR) counts for the ratio of the agent stopping at any position within 3 meters of the ground-truth goal, and is taken as the primary metric. (2) *Navigation Error* (NE) quantitatively characterizes the error between the final position and the goal by their shortest-path distance (in meters). (3) *Trajectory Length* (TL) computes the total length (in meters) of the agent's trajectory. (4) *Oracle Success Rate* (OSR) is an upper-bound for SR computed with an oracle stopping-rule. (5) *Success rate weighted by Path Length* (SPL) discounts the SR by considering the trajectory length.

Our Evaluated Models. As the shortest-path prior holds on R2R, we evaluate the classification model from Section 3.1 with both map known and unknown. Since most previous methods do not assume known maps, our comparison is mainly based on the unknown-map model, and only include the known-map model as an upper-bound.

Quantitative Result and Analysis. The results of our method with and without the map known on R2R are shown in Table 1, together with other competing approaches under both the single-run and beam-search (ensemble) settings. By default, our method explores up to 40 nodes in the environment when the map is unknown. Several conclusions could be drawn from Table 1.

First, regarding the primary metric SR, our method shows great superiority on both `val-unseen` and `test-unseen`, and beats most competitors on `val-seen`. Particularly, when combined with a sole Speaker [8] model following the convention, our model could even out-perform the three-model ensembled VLN-BERT [22] which needs pretraining with large amount of data, demonstrating the great effectiveness of harnessing the route prior. Furthermore, comparing the results on `val-seen` and `val-unseen` / `test-unseen`, most methods except for ours show large performance drop when tested on unseen environments, which implies that they are prone to overfitting to the training scenes to promote the results on `val-seen`. Instead, our model exhibits similar performance on all three sets, demonstrating its good generalization ability. Besides, the TL of our methods are shorter than others, thanks to the shortest-path prior we utilize, which also contributes to the high SPL. However, we also notice that the our NEs are worse than several methods despite higher SR. We believe this is one drawback of the classification model: an erroneous classification could induce large deviation from the goal. This is different from previous RL-based agents, where one decision only moves the agent one step, so even when the route is deviated by some choice, the agent could still navigate to the goal through later rectifications.

Table 1: Results of different methods on R2R dataset. All methods are trained with the augmented data from back-translation. ”-” means number not available. +spk means conducting model ensemble with a Speaker [8] model. Scores in bold and blue show the best and second best results in their columns, respectively.

Models	R2R														
	val-seen					val-unseen					test-unseen				
	SR↑	NE↓	TL↓	OSR↑	SPL↑	**SR**↑	NE↓	TL↓	OSR↑	SPL↑	**SR**↑	NE↓	TL↓	OSR↑	SPL↑
single-run															
Speaker-Follower [8]	0.66	3.36	-	0.74	-	0.36	6.62	-	0.45	-	0.35	6.62	14.8	0.44	0.28
RCM [29]	0.67	3.53	10.7	0.75	-	0.43	6.09	11.5	0.50	-	0.43	6.12	12.0	0.50	0.38
Self-Monitoring [20]	0.67	3.22	-	0.78	0.58	0.45	5.52	-	0.56	0.32	0.43	5.99	18.0	0.55	0.32
Tactical Rewind [16]	-	-	-	-	-	0.56	4.97	21.2	-	0.43	0.54	5.14	22.1	0.64	0.41
EnvDrop [25]	0.62	3.99	11.0	-	0.59	0.52	5.22	10.7	-	0.48	0.51	5.23	11.7	0.59	0.47
AuxRN [32]	0.70	3.33	-	0.78	0.67	0.55	5.28	-	0.62	0.50	0.55	5.15	-	0.62	0.51
Active Perception [27]	0.70	3.20	19.7	**0.80**	0.52	0.58	4.36	20.6	0.70	0.40	0.60	4.33	21.6	0.71	0.41
VLN-BERT [22]	0.70	3.73	10.3	0.76	0.66	0.59	4.10	9.6	0.69	0.55	-	-	-	-	-
SSM [26]	0.71	**3.10**	14.7	**0.80**	0.62	0.62	4.32	20.7	0.73	0.45	0.61	4.57	20.4	0.70	0.46
Ours(known map)	**0.76**	3.69	9.7	0.79	**0.72**	**0.74**	**3.56**	9.1	**0.80**	**0.71**	**0.72**	**3.77**	9.4	**0.76**	**0.69**
Ours(explore@40)	0.74	3.72	**9.6**	0.77	0.71	0.73	3.61	**9.0**	0.78	0.70	0.71	3.81	**9.3**	0.75	**0.69**
beam-search (ensemble) based															
Speaker-Follower [8]+spk	0.70	3.08	-	0.78	-	0.55	4.83	-	0.65	-	0.54	4.87	11.6	0.64	-
RCM [29]+spk	-	-	-	-	-	-	-	-	-	-	0.63	4.03	-	-	-
Self-Monitoring [20]+spk	0.71	3.04	-	0.78	0.67	0.58	4.62	-	0.68	0.52	0.61	4.48	-	0.70	0.56
Tactical Rewind [16]+spk	0.70	3.13	-	-	-	0.63	4.03	-	-	-	0.61	4.29	-	-	-
EnvDrop [25]+spk	0.76	2.52	-	-	-	0.69	3.08	-	-	-	0.69	3.26	-	-	-
AuxRN [32]+spk	-	-	-	-	-	-	-	-	-	-	0.71	3.24	-	-	-
Active Perception [27]+spk	-	-	-	-	-	-	-	-	-	-	0.70	**3.07**	-	-	-
VLN-BERT [22]+spk+flw	**0.82**	**2.35**	10.6	**0.87**	**0.78**	0.74	**2.76**	10.0	0.82	0.68	0.73	3.09	-	-	-
SSM [26]+spk	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Ours(known map)+spk	0.80	3.20	9.8	0.83	0.77	**0.79**	3.05	9.3	**0.85**	**0.76**	**0.76**	3.47	9.5	**0.79**	**0.73**
Ours(explore@40)+spk	0.78	3.27	**9.7**	0.81	0.75	0.78	3.13	**9.1**	0.83	0.74	0.74	3.55	**9.4**	0.78	0.72

Table 2: R2R online leaderboard results. We omit the results of the single-run models since they are exactly the same as Table 1. ”-” means number not available. +spk means conducting model ensemble with a Speaker [8] model. Numbers in bold and blue show the best and second best results in their columns.

Models	R2R				
	test-unseen				
	SR↑	NE↓	TL↓	OSR↑	SPL↑
beam-search (ensemble) based					
Speaker-Follower [8]+spk	0.54	4.87	1257.4	0.96	0.01
RCM [29]+spk	0.63	4.03	357.6	0.96	0.02
Self-Monitoring [20]+spk	0.61	4.48	373.1	0.97	0.02
Tactical Rewind [16]+spk	0.61	4.29	196.5	0.90	0.03
EnvDrop [25]+spk	0.69	3.26	686.8	0.99	0.01
AuxRN [32]+spk	0.71	3.24	**40.8**	0.81	**0.21**
Active Perception [27]+spk	0.70	**3.07**	176.2	0.94	0.05
VLN-BERT [22]+spk+flw	0.73	3.09	686.6	0.99	0.01
SSM [26]+spk	-	-	-	-	-
Ours(known map)	0.72	3.77	533.6	**1.00**	0.02
Ours(known map)+spk	**0.76**	3.47	533.7	**1.00**	0.02
Ours(explore@40)	0.71	3.81	625.1	0.99	0.01
Ours(explore@40)+spk	0.74	3.55	625.3	0.99	0.01

Lastly, comparing among all our methods, it can be seen that utilizing the known map is almost always better than exploring a sub-graph. There are mainly two reasons: (1) With the full map, the ground-truth route is assured to be in the candidates, while may be out of the span of the explored sub-graph. (2) The neighboring score aggregation functions only approximately on the sub-graph since it is incomplete, and we show in Section 4.3 that it may even hamper the performance if the sub-graph is too small.

Leaderboard Submission Results For fairly benchmarking our method, we follow the previous conventions [8, 22] to submit our results to the online server[1]. In Table 2, we include all necessary trajectories for exploration in the submission. For the known-map model, we run a Depth-First Search (DFS) to explore the whole graph and return back to the starting point. For the sub-graph based model, we just append the trajectory of the exploration agent at the beginning.

The main difference of Table 2 with 1 is that the TLs now are all long. But we still notice that our methods achieve the best SR with adequate TL with others, particularly with the current state-of-the-art VLN-BERT [22], justifying the effectiveness. Interestingly, we find that the TL of exploring the full map by DFS is even smaller than exploring the sub-graph of size 40. The latter conducts the State-Factored Search (SFF) [8] which involves abundant transfer within the already-explored graph. This implies that for environments with moderate size like those in R2R, conducting simple DFS is more preferred. But with larger environments, SFF is a better choice as it maintains a stable TL.

4.2. Results on R4R Dataset

Dataset. Room-for-Room (R4R) [15] is an extended dataset based on R2R. It connects a pair of paths head-by-

[1] https://eval.ai/web/challenges/challenge-page/97/leaderboard

tail (with necessary connection path) and joins their instructions to form a long instruction-path pair, which breaks the shortest-path prior and challenges more for fidelity to the instruction. R4R contains three splits: `train` (61 environments, 233,613 instructions), `val-seen` (56 environments, 1,035 instructions), and `val-unseen` (11 environments, 45,162 instructions) set. The annotation of the test set of R2R is kept private so is not included.

Evaluation Metric. Following [15, 14, 6, 26], we keep the SR, TL, and NE from R2R, and introduce three new metrics: (1) *Coverage weighted by Length Score* (CLS) measures the coverage of the model trajectory w.r.t. the ground-truth path. (2) *normalized Dynamic Time Warping* (nDTW) takes order-consistency into consideration, and (3) *Success rate weighted normalized Dynamic Time Warping* (SDTW) multiplies the SR with nDTW. The primary metric is CLS now.

Our Evaluated Models. For R4R, the shortest-path prior is no longer valid, so we only assume the general route prior and evaluate the sequential-decision model from Section 3.2 with both map known and unknown. Similar to above, the primary comparison is based on the unknown-map model.

Quantitative Result and Analysis. The results on R4R dataset are shown in Table 3. Following the conventions [15, 26], we only compare under the single-run setting. Firstly, our methods show surpassing CLS, which demonstrates the great fidelity to instructions. The similar results of nDTW further prove that our agent could find the ground-truth route with correct order. Regarding SR, although our model performs shyly on `val-seen`, it beats all other methods on `val-unseen` with clear margins. Similar to R2R, the large gap of other methods between `val-seen` and `val-unseen` shows that they overfit severely to the seen environments, while ours are more overfitting-free and rely mostly on the generalization ability. Lastly, comparing between our models, the difference is subtle with the known-map model slightly better, which is consistent with R2R.

4.3. Ablation Study of Key Components

To showcase the impact of several key design choices to the final performance, we conduct ablative studies on each of them to provide a comprehensive understanding of our method.

Size of the Explored Sub-Graph. In Section 4.1 and 4.2, we set the size of the explored sub-graph to 40 nodes by default. Here we provide a full list of results with various sub-graph sizes in Figure 3 for both R2R and R4R. It is clear that by increasing the size of the explored graph, the performance grows better and approaches the result of full map. Empirically, we find that setting the graph-size to 40 (nodes) strikes a good balance between the performance and the cost.

Neighboring Score Aggregation. The operation of neighboring score aggregation helps bridge the gap between the accuracy and SR, and here we compare the results with and without the aggregation in the first panel of Figure 3 on R2R. Note for R4R, the sequential model could not compute the exact probability of each node being the goal since it needs to sum over all possible paths, so we skip the aggregation directly. From Figure 3 we can see that when the graph-size is small, *e.g.*, 10 nodes, conducting aggregation actually hampers the performance, and when the graph-size grows larger (≥ 20), it instead contributes to the result. The reason is that as the topology of the sub-graph is incomplete, the aggregation may either be imprecise or unbalanced. As the graph-size increases, the topology is more close to the full map so the aggregation brings more benefit.

Progress-Aware Scoring Function. In Section 3.2, we propose to replace the MLP scoring function with a progress-aware module to respect the progress of partial paths on R4R. Here we quantitatively characterize its advantage in the left panel of Table 4. It is clear that using $\text{Aggr}_{\textbf{soft}}$ is indeed superior than MLP, but we also notice that even with the sub-optimal MLP function, the model still out-performs previous methods in Table 3, demonstrating the effectiveness of utilizing the route prior.

Solely the Exploration Agent. The explore-and-exploit scheme re-purposes an off-the-shelf VLN agent to build the sub-graph for our model. Here we compare the result with solely the exploration agent (the Follower) to demonstrate that the superiority indeed relies on our model. The results are shown in the right panel of Table 4. It is clear that solely the Follower model is far worse than our model. Note in Table 4 we count all the exploration trajectory in TL for fair comparison.

Combining with Single-Run Exploration-Agent. Lastly, we show the generalizability of our model by replacing the Follower model with a state-of-the-art single-run agent, SSM [26], for exploration. As shown in the right panel of Table 4, our method could still improve the performance based on purely the single-run explored graph of SSM which is far smaller than that of the beam-search agent.

5. Concluding Remarks

This work addresses the task of VLN from a previously-ignored aspect, namely the spatial route prior of the navigation scenes. For the most information-rich case of knowing environment maps and admitting shortest-path prior, we transform the route-finding problem into a node-classification task and design a straightforward model for it. Then we relax the route prior to a general case with a sequential-decision model and remove the map-awareness assumption with an explore-and-exploit scheme to fit our

Table 3: Results of different methods on R4R dataset. ”-” means number not available. Scores in bold and blue show the best and second best results in their columns, respectively.

Models	R4R											
	val-seen						val-unseen					
	SR↑	NE↓	TL	**CLS↑**	nDTW↑	SDTW↑	SR↑	NE↓	TL	**CLS↑**	nDTW↑	SDTW↑
Speaker-Follower [8]	0.52	5.35	15.4	0.46	-	-	0.24	8.47	19.9	0.30	-	-
RCM(goal oriented) [15]	0.56	5.11	24.5	0.40	-	-	0.29	8.45	32.5	0.20	0.27	0.11
RCM(fidelity oriented) [15]	0.53	5.37	18.8	0.55	-	-	0.26	8.08	28.5	0.35	0.30	0.13
PTA(low-level) [17]	0.57	5.11	11.9	0.52	0.42	0.29	0.27	8.19	10.2	0.35	0.20	0.08
PTA(high-level) [17]	0.58	**4.54**	16.5	0.60	**0.58**	0.41	0.24	8.25	17.7	0.37	0.32	0.10
EGP [6]	-	-	-	-	-	-	0.30	8.00	18.3	0.44	0.37	0.18
EnvDrop [25]	0.52	-	19.9	0.53	-	0.27	0.29	-	27.0	0.34	-	0.09
OAAM [24]	0.56	-	11.8	0.54	-	0.32	0.31	-	13.8	0.40	-	0.11
SSM [26]	**0.63**	4.60	19.4	0.65	0.56	**0.44**	0.32	8.27	22.1	0.53	0.39	0.19
Ours(known map)	0.45	6.92	20.1	**0.67**	0.57	0.38	**0.36**	7.46	20.5	**0.61**	**0.50**	**0.28**
Ours(explore@40)	0.45	6.35	19.8	0.66	**0.58**	0.36	**0.36**	**7.02**	20.2	0.60	**0.50**	0.27

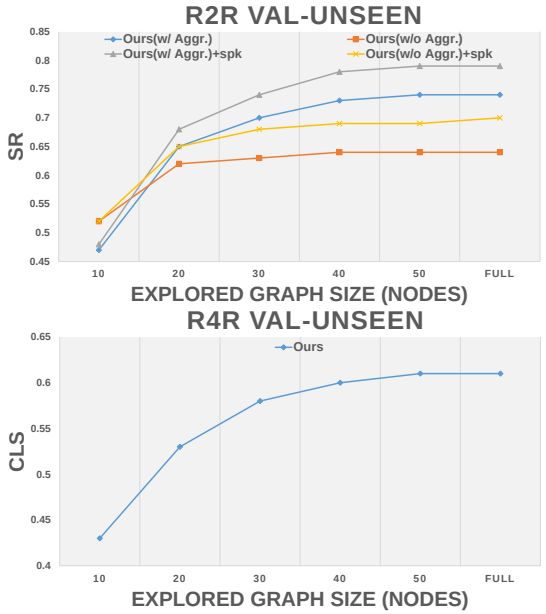

Figure 3: The full list of results by changing the size of the explored-graph on R2R (top) and R4R (bottom). +spk means conducting model ensemble with a Speaker [8] model.

model to more realistic situations. Extensive experiments and ablations on R2R and R4R clearly demonstrate the effectiveness of our design. Overall, we hope our work could help inspire new thoughts in VLN to further promote the development.

Table 4: Ablation study of the progress-aware scoring function (top) and the exploration agent (bottom). +spk means conducting model ensemble with a Speaker [8] model. Numbers in bold and blue show the best and second best results in their columns.

(a)

Models	R4R val-unseen		
	SR↑	NE↓	**CLS↑**
Ours(known map) MLP	0.32	8.38	0.58
Ours(known map) $\text{Aggr}_{\text{soft}}$	**0.36**	**7.46**	**0.61**
Ours(explore@40) MLP	0.33	7.59	0.58
Ours(explore@40) $\text{Aggr}_{\text{soft}}$	**0.36**	**7.02**	**0.60**

(b)

Models	R2R val-unseen		
	SR↑	NE↓	TL↓
beam-search			
Follower(explore@40)	0.39	6.32	636.3
Follower(explore@40)+spk	0.58	4.32	636.6
Ours(explore@40)	0.73	3.61	**636.0**
Ours(explore@40)+spk	**0.78**	**3.13**	636.1
single-run			
SSM	0.60	4.58	**45.4**
SSM + Ours	0.61	4.22	47.1
SSM + Ours + spk	**0.63**	**4.11**	47.0

References

[1] Peter Anderson, Angel X. Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, and Amir Roshan Zamir. On evaluation of embodied navigation agents. *CoRR*, abs/1807.06757, 2018.

[2] Peter Anderson, Ayush Shrivastava, Devi Parikh, Dhruv Batra, and Stefan Lee. Chasing ghosts: Instruction following as bayesian state tracking. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 369–379, 2019.

[3] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian D. Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 3674–3683, 2018.

[4] Howard Chen, Alane Suhr, Dipendra Misra, Noah Snavely, and Yoav Artzi. TOUCHDOWN: natural language navigation and spatial reasoning in visual street environments. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 12538–12547, 2019.

[5] Marco Cuturi and Mathieu Blondel. Soft-dtw: a differentiable loss function for time-series. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 894–903, 2017.

[6] Zhiwei Deng, Karthik Narasimhan, and Olga Russakovsky. Evolving graphical planner: Contextual global planning for vision-and-language navigation. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[7] Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[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 *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 3318–3329, 2018.

[9] Tsu-Jui Fu, Xin Eric Wang, Matthew F. Peterson, Scott T. Grafton, Miguel P. Eckstein, and William Yang Wang. Counterfactual vision-and-language navigation via adversarial path sampler. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part VI*, pages 71–86, 2020.

[10] 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 *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 13134–13143, 2020.

[11] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, 1997.

[12] Yicong Hong, Cristian Rodriguez Opazo, Yuankai Qi, Qi Wu, and Stephen Gould. Language and visual entity relationship graph for agent navigation. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[13] Haoshuo Huang, Vihan Jain, Harsh Mehta, Alexander Ku, Gabriel Magalhães, Jason Baldridge, and Eugene Ie. Transferable representation learning in vision-and-language navigation. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 7403–7412, 2019.

[14] Gabriel Ilharco, Vihan Jain, Alexander Ku, Eugene Ie, and Jason Baldridge. General evaluation for instruction conditioned navigation using dynamic time warping. In *Visually Grounded Interaction and Language (ViGIL), NeurIPS 2019 Workshop, Vancouver, Canada, December 13, 2019*, 2019.

[15] Vihan Jain, Gabriel Magalhães, Alexander Ku, Ashish Vaswani, Eugene Ie, and Jason Baldridge. Stay on the path: Instruction fidelity in vision-and-language navigation. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 1862–1872, 2019.

[16] Liyiming Ke, Xiujun Li, Yonatan Bisk, Ari Holtzman, Zhe Gan, Jingjing Liu, Jianfeng Gao, Yejin Choi, and Siddhartha S. Srinivasa. Tactical rewind: Self-correction via backtracking in vision-and-language navigation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 6741–6749, 2019.

[17] Federico Landi, Lorenzo Baraldi, Marcella Cornia, Massimiliano Corsini, and Rita Cucchiara. Perceive, transform, and act: Multi-modal attention networks for vision-and-language navigation. *CoRR*, abs/1911.12377, 2019.

[18] Jialu Li, Hao Tan, and Mohit Bansal. Improving cross-modal alignment in vision language navigation via syntactic information. *CoRR*, abs/2104.09580, 2021.

[19] Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1412–1421, 2015.

[20] 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 *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[21] Chih-Yao Ma, Zuxuan Wu, Ghassan AlRegib, Caiming Xiong, and Zsolt Kira. The regretful agent: Heuristic-aided navigation through progress estimation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 6732–6740, 2019.

[22] 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 *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part VI*, pages 259–274, 2020.

[23] Amin Parvaneh, Ehsan Abbasnejad, Damien Teney, Qinfeng Shi, and Anton van den Hengel. Counterfactual vision-and-language navigation: Unravelling the unseen. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

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

[25] Hao Tan, Licheng Yu, and Mohit Bansal. 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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 2610–2621, 2019.

[26] Hanqing Wang, Wenguan Wang, Wei Liang, Caiming Xiong, and Jianbing Shen. Structured scene memory for vision-language navigation. *CoRR*, abs/2103.03454, 2021.

[27] Hanqing Wang, Wenguan Wang, Tianmin Shu, Wei Liang, and Jianbing Shen. Active visual information gathering for vision-language navigation. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXII*, pages 307–322, 2020.

[28] Hu Wang, Qi Wu, and Chunhua Shen. Soft expert reward learning for vision-and-language navigation. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part IX*, pages 126–141, 2020.

[29] 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 *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 6629–6638, 2019.

[30] 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 *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XVI*, pages 38–55, 2018.

[31] Xin Eric Wang, Vihan Jain, Eugene Ie, William Yang Wang, Zornitsa Kozareva, and Sujith Ravi. Environment-agnostic multitask learning for natural language grounded navigation. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXIV*, pages 413–430, 2020.

[32] Fengda Zhu, Yi Zhu, Xiaojun Chang, and Xiaodan Liang. Vision-language navigation with self-supervised auxiliary reasoning tasks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 10009–10019, 2020.

[33] Wanrong Zhu, Yuankai Qi, Pradyumna Narayana, Kazoo Sone, Sugato Basu, Xin Eric Wang, Qi Wu, Miguel P. Eckstein, and William Yang Wang. Diagnosing vision-and-language navigation: What really matters. *CoRR*, abs/2103.16561, 2021.