# Robust Navigation with Language Pretraining and Stochastic Sampling

**Xiujun Li**♠◇  **Chunyuan Li**◇  **Qiaolin Xia**♣  **Yonatan Bisk**♠◇♡
**Asli Celikyilmaz**◇  **Jianfeng Gao**◇  **Noah A. Smith**♠♡  **Yejin Choi**♠♡

♠Paul G. Allen School of Computer Science & Engineering, University of Washington
♣Peking University  ◇Microsoft Research AI  ♡Allen Institute for Artificial Intelligence
{xiujun,ybisk,nasmith,yejin}@cs.washington.edu
xql@pku.edu.cn  {xiul,chunyl,jfgao}@microsoft.com

## Abstract

Core to the vision-and-language navigation (VLN) challenge is building robust instruction representations and action decoding schemes, which can generalize well to previously unseen instructions and environments. In this paper, we report two simple but highly effective methods to address these challenges and lead to a new state-of-the-art performance. First, we adapt large-scale pretrained language models to learn text representations that generalize better to previously unseen instructions. Second, we propose a stochastic sampling scheme to reduce the considerable gap between the expert actions in training and sampled actions in test, so that the agent can learn to correct its own mistakes during long sequential action decoding. Combining the two techniques, we achieve a new state of the art on the Room-to-Room benchmark with 6% absolute gain over the previous best result (47% $\rightarrow$ 53%) on the *Success Rate weighted by Path Length* metric.

## 1 Introduction

The vision-and-language navigation (VLN) task, learning to navigate in visual environments based on natural language instructions, has attracted interest throughout the artificial intelligence research community (Hemachandra et al., 2015; Anderson et al., 2018; Chen et al., 2019; Savva et al., 2019). It fosters research on multimodal representations and reinforcement learning, and serves as a test bed for many real-world applications such as in-home robots.

In the recent Room-to-Room (R2R) VLN challenge (Anderson et al., 2018), most state-of-the-art methods are developed based on an encoder-decoder framework (Cho et al., 2014; Sutskever et al., 2014), where a natural language instruction is represented as a sequence of words, and a navigation trajectory as a sequence of actions,
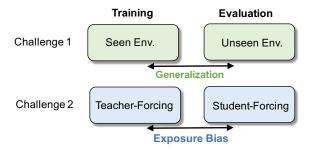


Figure 1: Two challenges in VLN.

enhanced with attention (Anderson et al., 2018; Wang et al., 2019; Fried et al., 2018; Ma et al., 2019a). Two important components are shared by all VLN agents: (i) an *Instruction Encoder* that employs a language model (LM) for instruction understanding; and (ii) an *Action Decoder*, where an appropriate sequence-level training scheme is required for sequential decision-making. Each component faces its own challenges (see Figure 1).

The first challenge is generalizing grounded natural language instruction understanding from seen to unseen environments. Specifically, in the R2R task, only 69% of bigrams are shared between training and evaluation.[1] Existing work leverages pretrained GloVe embeddings (Pennington et al., 2014) to help generalize. In computer vision, it has been shown that large-scale models pretrained on ImageNet can transfer the knowledge to downstream applications (Yosinski et al., 2014), thus improving generalization. Comparable language-based transfer learning has not been shown for instruction understanding in VLN.

The second challenge is *exposure bias* (Ranzato et al., 2016) for the action decoder, due to the discrepancy between training and inference. This problem is common to many tasks where decoding is needed, including text generation, abstractive summarization, and machine translation (Ben-

---

[1]Table 1 shows $n$-gram overlap statistics between training seen and validation seen/unseen environments.

| | n-gram(s) | Validation Seen | Validation Unseen |
|---|---|---|---|
| Training | 1 | 87.2% | 80.7% |
| | 2 | 77.4% | 68.9% |
| | 3 | 65.6% | 57.3% |
| | 4 | 50.8% | 44.4% |

Table 1: N-grams instruction overlap statistics between validation seen and unseen environments.



Figure 2: Illustration of proposed methods.

gio et al., 2015). Two widely used training strategies are *student-forcing* and *teacher-forcing* (described in detail in Section 2.2). It is well-known that the sequence length determines which training strategy is more effective. In the VLN literature, student-forcing has been widely used, as early work (Anderson et al., 2018) used long trajectories (up to 20 steps) with a simple discrete action space. Most recent work, however, has relied on a panoramic action space (Fried et al., 2018) in which most trajectories are only up to seven steps long. In such cases, teacher-forcing is preferable (Tan et al., 2019). Neither strategy is perfect: teacher-forcing has exposure bias, while student-forcing's random actions can cause an agent to deviate far from the correct path, rendering the original instruction invalid.[2]

To tackle these challenges, we have developed two techniques to enable the agent to navigate more efficiently. For the first challenge, we leverage the recent large-scale pretrained language models, BERT (Devlin et al., 2019) and GPT (Radford et al., 2018), to improve the agent's robustness in unseen environments. We show that large-scale *language-only* pretraining improves generalization in grounded environments. For the second challenge, we propose a stochastic sampling scheme to balance teacher-forcing and student-forcing during training, so that the agent can recover from its own mistakes at inference time. As a result of combining both techniques, on the R2R benchmark test set, our agent (PRESS)[3] achieves 53% on SPL, an absolute 6% gain over the current state of the art.

## 2 Method

In the VLN task, instructions are represented as a set $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^{M}$ of $M$ instructions per trajectory.
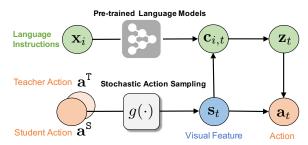
---

[2]To compensate, beam search is often used to improve success rates. Recent work, e.g., using search strategies (Ke et al., 2019) or progress monitors (Ma et al., 2019b), has focused on mitigating the cost of computing top-$k$ rollouts.

[3]PRETRAINED LMS AND STOCHASTIC SAMPLING

Each instruction $\boldsymbol{x}_i$ is a sequence of $L_i$ words, $\boldsymbol{x}_i = [x_{i,1}, x_{i,2}, ..., x_{i,L_i}]$. Given $\mathcal{X}$, the goal is to train an agent to navigate from a starting position $\boldsymbol{s}_0$ to a target position, via completing a $T$-step trajectory $\tau = [\boldsymbol{s}_0, \boldsymbol{a}_0, \boldsymbol{s}_1, \boldsymbol{a}_1, \cdots, \boldsymbol{s}_T, \boldsymbol{a}_T]$, where $\boldsymbol{s}_t$ and $\boldsymbol{a}_t$ are the visual state and navigation action, respectively, at step $t$. The training dataset $\mathcal{D}_E = \{\tau, \mathcal{X}\}$ consists of example pairs of instruction set $\mathcal{X}$ and a corresponding expert trajectory $\tau$. Our goal is to learn a policy $\pi_{\boldsymbol{\theta}}(\tau|\mathcal{X})$ that maximizes the log-likelihood of the target trajectory $\tau$ given instructions $\mathcal{X}$:

$$\log \pi_{\boldsymbol{\theta}}(\tau|\mathcal{X}) = \sum_{t=1}^{T} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t|\boldsymbol{s}_t, \mathcal{X}), \quad (1)$$

where $\boldsymbol{\theta}$ are trainable parameters. The policy is usually parameterized as an attention-based seq2seq model, with a language encoder $\boldsymbol{z}_t = f_{\theta_E}(\boldsymbol{x})$, and an action decoder $\boldsymbol{a}_t = f_{\theta_D}(\boldsymbol{z}_t, \boldsymbol{s}_t)$. Successful navigation depends on ($i$) precisely grounding the instructions $\mathcal{X}$ in $\tau$ in various environments, and ($ii$) correctly making the current decision $\boldsymbol{a}_t$ based on previous actions/observations $\tau_{<t} = [\boldsymbol{s}_0, \boldsymbol{a}_0, \cdots, \boldsymbol{s}_{t-1}]$. To address these concerns, we propose PRESS, illustrated in Figure 2.

### 2.1 Instruction Understanding with Pretrained Language Models

At each step $t$, the agent decides where to navigate by updating a dynamic understanding of the instructions $\boldsymbol{z}_t$, according to its current visual state $\boldsymbol{s}_t$. Given instruction $\boldsymbol{x}$, the language encoder proceeds in two steps, end-to-end, by considering a function decomposition $f_{\theta_E} = f_{\boldsymbol{\theta}_{x \to e}} \circ f_{\boldsymbol{\theta}_{e \to z}}$:

- $f_{\boldsymbol{\theta}_{x \to e}} : \boldsymbol{x} \to \boldsymbol{e}$, where $\boldsymbol{x} = [x_1, \cdots, x_L]$ is represented as its (contextualized) word embedding form $\boldsymbol{e} = [e_1, \cdots, e_L]$, with $e_i$ as the representation for word $x_i$;

- $f_{\boldsymbol{\theta}_{e \to z}} : \boldsymbol{e} \to \boldsymbol{z}_t$: For each embedded instruction $\boldsymbol{e}$, we ground its representations as $\boldsymbol{c}_{i,t}$ for state $\boldsymbol{s}_t$ via neural attention. To handle

language variability, one may aggregate features of multiple instructions $\mathcal{C}_t = \{c_{i,t}\}_{i=1}^{M}$ into a single joint feature $z_t = \frac{1}{M}\sum_{i=1}^{M} c_{i,t}$.[4]

Previous methods in VLN learn $e$ either from pretrained word embeddings (Pennington et al., 2014) which do not take into account word context, or from scratch. As a result, their representations do not capture contextual information within each instruction. More importantly, they tend to overfit the training instructions associated with seen environments, limiting their utility in unseen environments. To remedy these issues, we propose to represent $e$ with contextualized word embeddings produced using large-scale pretrained language models, such as BERT and GPT.

**Instruction Encoder.** The agent's memory vector $h_{t-1}$ captures the perception and action history and is used to attend to the instruction $x$. A pretrained LM $f_{\theta_{x\to e}}$ encodes the instruction $e = [e_1, \cdots, e_L]$; $e_i$ where the representation for word $x_i$, is built with $f_{\theta_{x\to e}} \in \{$ GPT, BERT $\}$, and $\theta_{x\to e}$ are fine-tuned parameters. The embedded words $e = [e_1, \cdots, e_L]$ are passed through an LSTM $f_{\theta_{e\to z}}$ to produce a sequence of textual features $[h_1^e, \cdots, h_L^e]$. At each time step $t$, the *textual context* for the instruction $x$ is computed as weighted sum of textual features in the sequence:

$$c_{i,t} = \sum_{l=1}^{L} \alpha_l h_l^e \qquad (2)$$

where $\alpha_l = \text{Softmax}(h_t^\top h_l^e)$, $\alpha_l$ places more weight on the word representations that are most relevant to the agent's current status.

**Decoder.** At each step, the agent takes an action $a_t$, and the environment returns new visual observations; the agent first performs one-hop visual attention $f(\cdot)$ to all the visual image features $s_t$, based on its previous memory vector $h_{t-1}$. Then, the agent updates its visual state $s_t$ as the weighted sum of the panoramic features, $s_t = \sum_j \gamma_{t,j} s_{t,j}$. The attention weight $\gamma_{t,j}$ for the $j$-th visual feature $s_{t,j}$ represents its importance with respect to the previous history context $h_{t-1}$, computed as $\gamma_{t,j} = \text{Softmax}((W_h h_{t-1})^\top W_s s_{t,j})$ (Fried et al., 2018) where $\text{Softmax}(r_j) = \exp(r_j)/\sum_{j'} \exp(r_{j'})$,

---

[4] This recovers $z_t = c_t$ when only a single instruction is available.

$W_h$ and $W_s$ are trainable projection matrices.

$$h_t = f_{\theta_D}([s_t, a_{t-1}], h_{t-1}) \qquad (3)$$

where $a_{t-1}$ is the action taken at previous step, and $\theta_D$ are the LSTM decoder parameters.

**Two-stage learning.** The parameters of our agent are $\theta = \{\theta_{x\to e}, \theta_{e\to z}, \theta_D\}$. In practice, we find that the agent overfits quickly, when the full model is naively fine-tuned, with $\theta_{x\to e}$ initialized by pretrained LMs (*e.g.,* BERT). In this paper, we consider a two-stage learning scheme to facilitate the use of pretrained LMs for VLN. (*i*) Embedding-based stage: We fix $\theta_{x\to e}$, and use BERT or GPT to provide instruction embeddings. Only $\{\theta_{e\to z}, \theta_D\}$ are updated (while tuning on validation). (*ii*) Fine-tuning stage: We train all model parameters $\theta$ with a smaller learning rate, so that $\theta_{x\to e}$ can adapt to our VLN task.

## 2.2 Stochastic Action Sampling

A core question is how to learn useful state representations $s_t$ in Eq. (1) during the trajectory rollout. In other words, which action should we use to interact with the environment to elicit the next state? As noted, most existing work uses one of two schemes: (*i*) *Teacher-forcing (TF)*, where the agent takes ground-truth actions $a^{\mathrm{T}}$ only. Though TF enables efficient training, it results in "exposure bias" because agents must follow learned rather than gold trajectories at test time. In contrast, (*ii*) *Student-forcing (SF)*, where an action $a^{\mathrm{S}}$ is drawn from the current learned policy, allows the agent to learn from its own actions (aligning training and evaluation), however, it is inefficient, as the agent explores randomly when confused or in the early stages of training.

In this work, we consider a stochastic scheme (*SS*) to alternate between choosing actions from $a^{\mathrm{T}}$ and $a^{\mathrm{S}}$ for state transition $s \leftarrow g(a^{\mathrm{T}}, a^{\mathrm{S}})$, inspired by scheduled sampling (Bengio et al., 2015). As illustrated in Figure 2, at each step, the agent "flips a coin" with some probability $\epsilon$ to decide whether to take the teacher's action $a^{\mathrm{T}}$ or a sampled one $a^{\mathrm{S}}$:

$$a = \delta a^{\mathrm{T}} + (1 - \delta) a^{\mathrm{S}}, \qquad (4)$$

where $\delta \sim \text{Bernoulli}(\epsilon)$. This allows the agent to leverage the advantages of both TF and SF, yielding a faster and less biased learner. We fix $\epsilon$ as a constant during learning, which is different from the decaying schedule in (Bengio et al., 2015).

# 3 Experiments

## 3.1 Dataset

We use the Room-to-Room dataset for the VLN task, built upon the Matterport3D dataset (Chang et al., 2017), which consists of 10,800 panoramic views and 7,189 trajectories. Each trajectory is paired with three natural language instructions. The R2R dataset consists of four splits: train seen, validation seen, validation unseen, and test unseen. There is no overlap between seen and unseen environments. At the beginning of each episode, the agent starts at a specific location, and is given natural instructions, the goal of the agent is to navigate to the target location as quickly as possible.

## 3.2 Baseline Systems

We compare our approach with eight recently published systems:

- RANDOM: an agent that randomly selects a direction and moves five step in that direction (Anderson et al., 2018).
- SEQ2SEQ: sequence-to-sequence model proposed by Anderson *et al.* as a baseline for the R2R benchmark (Anderson et al., 2018) and analyzed in (Thomason et al., 2019).
- RPA (Wang et al., 2018): is an agent which combines model-free and model-based reinforcement learning, using a look-ahead module for planning.
- SPEAKER-FOLLOWER (Fried et al., 2018): an agent trained with data augmentation from a speaker model with panoramic actions.
- SMNA (Ma et al., 2019a): an agent trained with a visual-textual co-grounding module and progress monitor on panoramic actions.
- RCM+SIL(TRAIN) (Wang et al., 2019): an agent trained with cross-modal grounding locally and globally via reinforcement learning.
- REGRETFUL (Ma et al., 2019b): an agent with a trained progress monitor heuristic for search that enables backtracking.
- FAST (Ke et al., 2019): an agent which combines global and local knowledge to compare partial trajectories of different lengths, enabling efficient backtrack after a mistake.
- ENVDROP (Tan et al., 2019): proposed an environment dropout method, which can generate more environments based on the limited seen environments.

| Setting | Agent | Validation Seen | | Validation Unseen | |
|---|---|---|---|---|---|
| | | SR ↑ | SPL ↑ | SR ↑ | SPL ↑ |
| S | seq2seq | 51 | 46 | 32 | 25 |
| | PRESS | 47 (-4) | 43 (-3) | 43 (+11) | 38 (+13) |
| M | seq2seq | 49 | 44 | 33 | 26 |
| | PRESS | 56 (+7) | 53 (+9) | 56 (+23) | 50 (+24) |

Table 2: Comparison of PRESS and seq2seq.

## 3.3 Evaluation Metrics

We benchmark our agent on the following metrics:

**TL Trajectory Length** measures the average length of the navigation trajectory.

**NE Navigation Error** is the mean of the shortest path distance in meters between the agent's final location and the target location.

**SR Success Rate** with which the agent's final location is less than 3 meters from the target.

**SPL Success weighted by Path Length** trades-off SR against TL.

SPL is the recommended primary metric, other metrics are considered as auxiliary measures.

## 3.4 Implementation

We use a LSTM/GPT/BERT for the language encoder, and a second single-layer LSTM for the action decoder (h=1024). We use Adamax and batch sizes of 24/16 for pretraining/finetuning. The learning rates for MLE are $1e^{-4}$, during finetuning BERT the learning rate is $5e^{-5}$. Following (Fried et al., 2018), we use a panoramic action space and the ResNet image features provided by (Anderson et al., 2018). The code is publicly available here: https://github.com/xjli/r2r_vln.

## 3.5 Results

**Robust Generalization.** First, we compare PRESS to a baseline seq2seq model[5] in two evaluation settings on the validation splits: (1) **S**: A single instruction is provided to the agent at a time. Thus, three separate navigation trajectories are generated corresponding to three alternative instructions in this setting. We report the averaged performance over three separate runs. (2) **M**: All three instructions are provided to the agent at once. The seq2seq baseline does not have an aggregation strategy so we report its performance for the single trajectory with maximum likelihood. For PRESS, we aggregate the instructions via context

---

[5]The baseline seq2seq agent is the FOLLOWER of SPEAKER-FOLLOWER (Fried et al., 2018).

| | Model | Validation Seen | | | | Validation Unseen | | | | Test Unseen | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | TL↓ | NE↓ | SR↑ | SPL↑ | TL↓ | NE↓ | SR↑ | SPL↑ | TL↓ | NE↓ | SR↑ | SPL↑ |
| | RANDOM | 9.58 | 9.45 | 16 | - | 9.77 | 9.23 | 16 | - | 9.93 | 9.77 | 13 | 12 |
| | SEQ2SEQ | 11.33 | 6.01 | 39 | - | 8.39 | 7.81 | 22 | - | 8.13 | 7.85 | 20 | 18 |
| | RPA | - | 5.56 | 43 | - | - | 7.65 | 25 | - | 9.15 | 7.53 | 25 | 23 |
| Greedy | SPEAKER-FOLLOWER | - | 3.36 | 66 | - | - | 6.62 | 35 | - | 14.82 | 6.62 | 35 | 28 |
| | SMNA | - | - | - | - | - | - | - | - | 18.04 | 5.67 | 48 | 35 |
| | RCM+SIL(TRAIN) | 10.65 | 3.53 | 67 | - | 11.46 | 6.09 | 43 | - | 11.97 | 6.12 | 43 | 38 |
| | REGRETFUL | - | 3.23 | 69 | 63 | - | 5.32 | 50 | 41 | 13.69 | 5.69 | 48 | 40 |
| | FAST | - | - | - | - | 21.17 | 4.97 | 56 | 43 | 22.08 | 5.14 | 54 | 41 |
| | ENVDROP | 11.00 | 3.99 | 62 | 59 | 10.70 | 5.22 | 52 | 48 | 11.66 | 5.23 | 51 | 47 |
| | PRESS | 10.35 | **3.09** | **71** | **67** | 10.06 | **4.31** | **59** | **55** | 10.52 | **4.53** | **57** | **53** |
| | Human | - | - | - | - | - | - | - | - | 11.85 | 1.61 | 86 | 76 |

Table 3: Comparison with the state-of-the-art methods. **Blue** indicates best value overall.

mean-pooling and generate a single trajectory. No data augmentation is applied to either model.

The results are summarized in Table 2. (*i*) PRESS drastically outperforms the seq2seq models on unseen environments in both settings, and (*ii*) Interestingly, our method shows a much smaller gap between seen and unseen environments than seq2seq. It demonstrates the importance of pretrained LMs and stochastic sampling for strong generalization in unseen environments.

**Comparison with SoTA.** In Table 3, we compare the performance of our agent against all the published methods, our PRESS agent outperforms the existing models on nearly all the metrics.

**Ablation Analysis.** Key to this work is leveraging large-scale pretrained LMs and effective training strategies for action sequence decoding. Table 4 shows an ablation of these choices. (1) BERT and GPT are better than LSTM on both seen and unseen environments, and BERT generalizes better than GPT on unseen environments. (2) Teacher-forcing performs better than student-forcing on validation unseen environments, while an opposite conclusion is drawn on validation seen environments. SS performs the best on unseen environments.

**Qualitative Examples.** We provide two navigation examples of PRESS on the validation unseen environments with the step-by-step views and top-down views in Appendix.

(1) Figure 3 shows how the agent with LSTM instruction encoder performs compared with our PRESS agent. There are two rare words "**mannequins**" and "**manikins**" which are not in the training dataset and confuse the LSTM agent, while, PRESS successfully maps these two "mannequins" and "manikins" to the correct objects.

| LM | | Validation Seen | | | | Validation Unseen | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | TL | NE | SR | SPL | TL | NE | SR | SPL |
| LSTM | TF | 10.50 | 5.74 | 44 | 42 | 9.86 | 6.23 | 42 | 39 |
| | SF | 11.87 | 3.97 | 59 | 53 | 13.23 | 6.17 | 40 | 31 |
| | SS | 10.99 | 3.46 | 64 | 59 | 10.73 | 4.89 | 53 | 48 |
| GPT | TF | 10.03 | 4.05 | 60 | 58 | 9.43 | 3.36 | 49 | 46 |
| | SF | 11.46 | 2.53 | 73 | 67 | 13.13 | 5.13 | 49 | 41 |
| | SS | 10.60 | 2.99 | 71 | 68 | 10.79 | 3.05 | 56 | 51 |
| BERT | TF | 10.57 | 4.06 | 59 | 56 | 9.61 | 5.13 | 51 | 47 |
| | SF | 12.39 | 2.71 | 73 | 64 | 13.12 | 5.06 | 51 | 42 |
| | SS | 10.35 | 3.09 | 71 | 67 | 10.06 | 4.31 | **59** | **55** |

Table 4: Ablation results of different language pretrainings and training strategies: Teacher Forcing (TF), Student Forcing (SF) and Stochastic Sampling (SS).

(2) The second set in Figure 4 shows how the agents trained with different training strategies performs in an unseen environment. The agents trained with teacher-forcing and student-forcing both fail, while PRESS succeeds.

## 4 Conclusion

We present PRESS, a navigation agent based on two previously underexplored techniques in VLN: pretrained language models and stochastic action sampling. Our PRESS demonstrates robust generalization in the unseen environments, leading to a new state-of-the-art performance over many of the much more complex approaches previously proposed. As both the components of PRESS can be easily integrated, future models can consider building upon them as a strong baseline system.

## Acknowledgments

## References

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. 2018. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Neural Information Processing Systems*.

Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Nießner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. 2017. Matterport3D: Learning from RGB-D data in indoor environments. In *International Conference on 3D Vision*.

Howard Chen, Alane Shur, Dipendra Misra, Noah Snavely, and Yoav Artzi. 2019. Touchdown: Natural language navigation and spatial reasoning in visual street environments. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. 2018. Speaker-follower models for vision-and-language navigation. In *Neural Information Processing Systems*.

Sachithra Hemachandra, Felix Duvallet, Thomas M Howard, Nicholas Roy, Anthony Stentz, and Matthew R Walter. 2015. Learning models for following natural language directions in unknown environments. In *IEEE International Conference on Robotics and Automation*.

Liyiming Ke, Xiujun Li, Yonatan Bisk, Ari Holtzman, Zhe Gan, Jingjing Liu, Jianfeng Gao, Yejin Choi, and Siddhartha Srinivasa. 2019. Tactical rewind: Self-correction via backtracking in vision-and-language navigation. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Chih-Yao Ma, Jiasen Lu, Zuxuan Wu, Ghassan Al-Regib, Zsolt Kira, Richard Socher, and Caiming Xiong. 2019a. Self-monitoring navigation agent via auxiliary progress estimation. In *International Conference on Learning Representations*.

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

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Conference on empirical methods in natural language processing*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *International Conference on Learning Representations*.

Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. 2019. Habitat: A platform for embodied ai research. In *International Conference on Computer Vision*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Neural Information Processing Systems*.

Hao Tan, Licheng Yu, and Mohit Bansal. 2019. Learning to navigate unseen environments: Back translation with environmental dropout. In *the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Jesse Thomason, Daniel Gordon, and Yonatan Bisk. 2019. Shifting the Baseline: Single Modality Performance on Visual Navigation & QA. In *the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

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

Xin Wang, Wenhan Xiong, Hongmin Wang, and William Yang Wang. 2018. Look before you leap: Bridging model-free and model-based reinforcement learning for planned-ahead vision-and-language navigation. In *IEEE European Conference on Computer Vision*.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? In *Neural Information Processing Systems*.

**Instruction A**: *Go up the stairs to the right, turn left and go into the room on the left. Turn left and stop near the* **mannequins**.

**Instruction B**: *Walk up the small set of stairs. Once you reach the top, turn 45 degrees to your left. Walk through the door at the bottom of the large staircase. After you are inside, turn left and wait near the statue.*

**Instruction C**: *Walk up the stairs Through the doorway on the left. Make a left in the room and stop before the two* **manikins**.
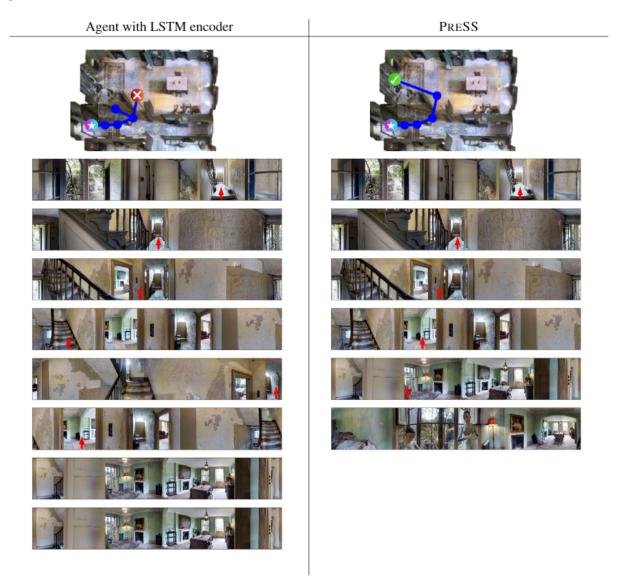


Figure 3: Comparison between the agent equipped with an LSTM instruction encoder and our PRESS agent on a validation unseen environment (path_id: 6632), including top-down trajectory view and step-by-step navigation views. We indicate the start (⭐), target (✅) and failure (❌) of agents in an unseen environment.

Figure 4: Comparison among the agents trained with teacher-forcing, student-forcing and stochastic sampling strategies on a validation unseen environment (path_id: 7201), including top-down trajectory view and step-by-step navigation views. We indicate the start (⬢), target (✓) and failure (✗) of agents in an unseen environment.