11-777 Vision Language Navigation R2R

Linwei Li [*1] **Yifan Zhou** [*1] **Yuqing Deng** [*1] **Zhongyan Lu** [*1]

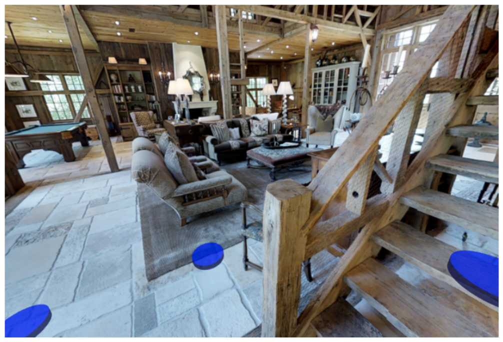
Abstract

The Room-to-Room challenge in the field of Vision-Language Navigation aims to have an autonomous agent that can interpret and carry out a detailed natural language instruction to navigate in an indoor real 3D environment. In this paper, we focus on how to solving these crucial research challenges: multi-modal joint representation learning, robust trajecory encoding and error correction. Our main ideas include adding an external memory module to encode trajectory information, applying contrastive learning to learn a better joint representation and utilizing DAGGER to rectify a deviated agent. Evaluation on the R2R validation datasets shows that the agent with contrastive learning outperforms the state-of-the-art PREVALENT agent on SR and SPL. Based on extensive experiments, we conclude that 1) External memory significantly reduces repeated navigation errors; 2) Contrastive learning leads to an interpretable and helpful joint representation; 3) Data aggregation can improve navigation accuracy and guide the agent back on track. [1]

1. Introduction

One of the most desired goal people have long hoped to achieve with Robotics and Artificial Intelligence is that given a general verbal instruction, a robot can carry out the task independently. In the area of navigation, autonomous agents such as household robots need to interact with the physical world in multiple modalities, mostly vision and language. Vision-and-Language Navigation (VLN) is the term of such a problem.

The Room-to-Room challenge and its dataset were introduced in 2018 (Anderson et al., 2018). An instruction was given prior to robot movements, and images from different angles and different directions can be acquired by the robot at the robot's current position. Given the instruction and a set of images, the robot should choose an action to navigate to a next nearby location. An example of a set of instruction, view images, and action options is shown in Figure 1.



Instruction: Head upstairs and walk past the piano through an archway directly in front. Turn right when the hallway ends at pictures and table. Wait by the moose antlers hanging on the wall.

Figure 1. Room-to-Room (R2R) navigation task. Blue discs indicate nearby (discretized) navigation options.

Our initial experiments and evaluations indicated that the prior multi-modal baseline models have not been able to fully address the following research challenges.

Multi-modal joint representation A crucial challenge faced by almost all vision-language multi-modal tasks is taking full advantage of the visual, language modalities and past trajectory information. At each timestep, the VLN agent is provided a set of images at the current location and an instruction that describes the past, current, and future trajectory. Learning the most accurate and informative joint representation of image, instruction and trajectory would enable the agent to perform better action prediction.

Robust trajectory encoding In the VLN task, the agent is required to memorize its past trajectories including the visited viewpoints and performed actions. The past trajectory is essential for the agent to locate itself in the instruction. Moreover, sufficient historical information can enable agent to avoid repeating the same mistake if it revisits some viewpoint. Also, an instruction can contain multiple occurrences of the same entities (for example, the agent is required to

[*]Equal contribution [1]Langauge Technologies Institute, School of Computer Science, Carnegie Mellon University, Pittsburgh, United States. Correspondence to: Yifan Zhou <yifanzh3@andrew.cmu.edu>.

[1]The source code of our project is located at https://github.com/yfzhoucs/VLN-R2R

walk through two doors sequentially), and the agent should be able to distinguish between them.

Error correction It is challenging for the agent to get back on track once it deviates from the correct route. Therefore, some explicit error correction mechanism can be useful to rectify the deviated agent.

In this paper, we propose three novel approaches to help resolve these challenges. Our major focus is on leveraging the past trajectory information and generating more informative trajectory encoding. The first approach is to supplement the model with trajectory by utilizing an external memory module instead of a naive LSTM. Our second approach is to apply contrastive learning with the task of predicting whether the agent is currently on-track to learn a better vision-language joint representation, and supply this joint representation to the agent's decoder in addition to the features representations. Finally, we propose to use DAGGER to generate ground truth actions for deviated agents in supervised training.

Section 2 presents an overview of the related works on the R2R challenges as well as external memory and DAGGER. Section 3 explains our three research ideas in detail. Section 4 describes our experiment settings. Section 5 reports and analyzes the experiment results. Section 6 summarizes our experiments and section 7 introduces some potential improvements and extensions.

2. Related Work

2.1. R2R Navigation Task

We begin by introducing some of the most relevant prior works on the R2R navigation task.

2.1.1. Training an action-predicting agent

A navigation agent is required to follow the instruction to generate a route from the start location to the target location. The earliest baseline for the Room-to-Room dataset (Anderson et al., 2018) is an encoder-decoder LSTM model with general attention mechanism. The language instruction is encoded in the first LSTM layer, as the context for attention, and the second layer takes in the image at each viewpoint of the route and outputs the corresponding action.

Afterwards, more works have extended the agent to have more complex architecture, such as recurrent policy network along with look-ahead module (Gruber et al., 2008), two-tower architecture of visual encoder and language encoder (Huang et al., 2019), and visual-textual co-grounding attention mechanism with progress monitor (Ma et al., 2019).

In terms of optimizing the agent, supervised learning methods can be applied given the ground truth action data. Imitation learning and reinforcement learning are also popularly used (Gruber et al., 2008; Tan et al., 2019; Wang et al., 2019).

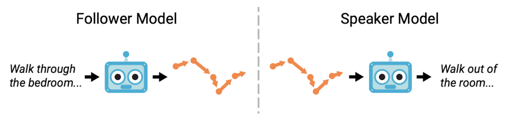


Figure 2. Illustration of the speaker-follower architecture.

2.1.2. Selecting path using Speaker-Follower

With an existing agent, another group of works focus on generating candidate routes and selecting a high-probability route. In the speaker-follower architecture (see Figure 2) (Fried et al., 2018), the follower uses the same architecture as the baseline sequence-to-sequence (Anderson et al., 2018) and can output routes from instructions. The speaker is also a two-layer LSTM that can generate instruction given the route. Beam search and state-factored search are performed to generate candidate routes through environment, and the routes are ranked based on how well they can be explained by the speaker.

2.1.3. Data Augmentation and Pretraining

A major challenge of this task is posed by the dataset's small size. Therefore, the speaker model in (Fried et al., 2018) is often used for data augmentation, since it can generate synthetic instructions for the routes.

Pre-training is also commonly used to compensate for the lack of training data and exploit the progress in other research areas. Image-text pairs crawled from the Web were used to train the path-instruction compatibility model in (Majumdar et al., 2020). PREVALENT (Hao et al., 2020) focused on pre-training an encoder using a simple model that takes in the entire instruction and the current vision (see Figure 3), while applying two key pre-training tasks to improve the model performance: image-attended masked language modeling, similar to BERT; and action prediction using the fused representation of two modalities.

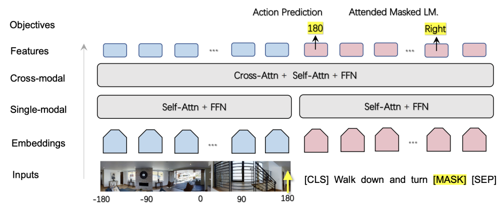


Figure 3. PREVELANT model architecture illustration.

2.2. External memory

To address the problem of insufficient past trajectory information, we look at prior works on explicit external memory modules. (Graves et al., 2016) proposed the Differentiable Neural Computer (DNC) architecture which was able to read from and write to an external memory matrix, and could be trained to perform inference tasks. Inspired by the DNC work, (Hill et al., 2020) came up with an external key-value memory architecture for vision and language agents, which could facilitate the agent to manipulate a novel object as instructed.

2.3. Dataset aggregation in imitation learning

A dataset aggregation algorithm (DAGGER) was proposed (Ross et al., 2011) with the main idea of collecting dataset at each training iteration and using the aggregate of collected dataset to train the next policy. This approach was proved to have robust performance on imitation learning tasks.

2.4. Differentiation of our ideas

To our knowledge, our first approach that uses external memory to encode trajectory distinguishes from all previous works as the existing models simply encode trajectory via LSTM or do not include trajectory in pre-training at all. The second approach (learning joint representation using contrastive learning) is the first one that use the on-trajectory information to do contrastive pre-training on joint representation. Finally, the DAGGER algorithm has so far never been used to assist the training of VLN agents.

3. Proposed Approach

3.1. External Memory

The state of the art works mostly use LSTMs to encode trajectory, which has limited capability of learning informative trajectories that help the agent choose better actions. In our experiments, we find some common error cases from the baseline model, such as wandering back and forth between viewpoints, which indicates the agent cannot leverage past trajectory which should contain the failed actions. Therefore, it is promising to strive for a better way of memorizing the past trajectory. We consider to use an external memory adapted from (Hill et al., 2020). The adopted memory remembers the image and text information at each timestep, which helps the agent avoid wrong actions if it was taken at the same viewpoint. The following paragraphs will illustrate how the external memory read and write information.

3.1.1. Write Operation

The memory is designed to be a list of multiple slots, each of which represents a timestep. Therefore, the length of the memory increases over time. The memory is cleared after the end of each episode. The write operation takes in 36 vision embeddings $f_{1...36}$ from the current viewpoint, as well as the hidden state vector h_t of the encoder LSTM which represents the text embedding. During writing, firstly the 36 vision embeddings are put through a cross-attention layer attended on text embeddings, which generates a vector embedding $\widetilde{f}$. Afterwards, this embedding is concatenated with the text embedding, together projected as the vector m_t to be appended to the memory slot.

$$m_t = \textbf{Linear}([\widetilde{f}, h_t])$$
$$\widetilde{f} = \sum_{i=0}^{36} a_i * f_i$$
$$a_1, a_2, ... = \textbf{softmax}(A_1, A_2, ...)$$
$$A_i = f_i^T W_{attn} \widetilde{h}_t$$

3.1.2. Read Operation

The read operation is also attentive. The output of the decoder is used as the query. Firstly, the query is fed into an attention layer, where the keys and values are both the memory slots. The output of the attention layer, the readout vector r_t, is then put through an LSTM inside the memory. Afterwards, the output of the LSTM is concatenated with the original query, and linearly projected onto the original dimension, forming the output o_t.

$$o_t = \textbf{Linear}([r_t, query])$$
$$r_t = \textbf{LSTM}(\textbf{attn}(query, m_{1...t}, m_{1...t}))$$

3.2. Contrastive Training for Alignment-Aware Joint Representations

Previous works encode the language and the vision separately and feed them to the agent in parallel. Although attention across modalities might be used, such fusion only happens in the later stage of the model and the fusion process itself is not separately trained (only trained along with the navigation task). In addition, in our preliminary experiments, we found that the attention map of the cross attention between the vision and the instruction is usually unclear, especially when the agent is in a confusing state. Unclear attention maps are often followed by a wrong move. Therefore, simply relying on attention mechanism to align two modalities is not sufficient. We propose to pretrain a joint representation through contrastive learning on alignment prediction tasks. We believe joint representations produced by this method is more beneficial to navigation tasks.

3.2.1. Training Procedure

A huge advantage of contrastive learning in our task is the data abundance due to self-supervised training. For each instruction, we sample a positive viewpoint which is on the correct trajectory. And we sample a negative

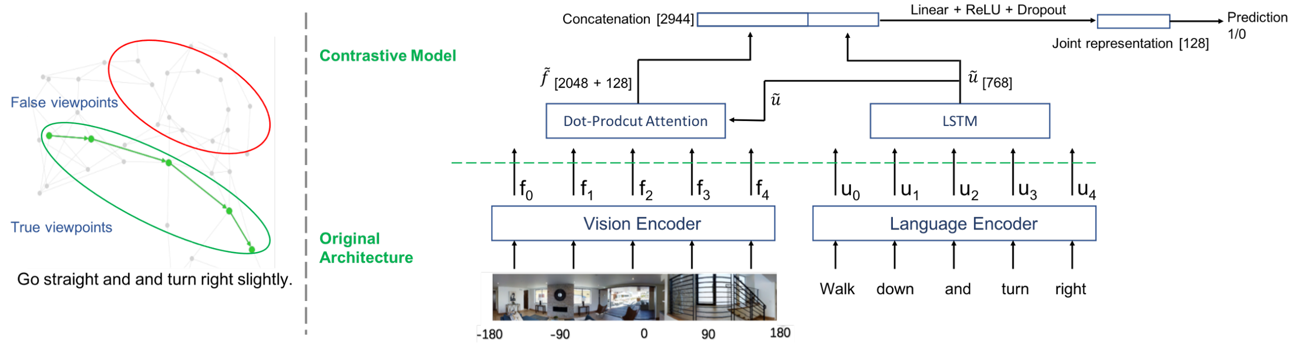


Figure 4. Contrastive Training. Illustration of the sampling process (left). Architecture for the contrastive model (right).

viewpoint randomly in the current scan (environment). The contrastive model predicts how well the viewpoint is on trajectory according to the instruction, in a scale from 1 to 0.

In later experiments, we only sample the wrong viewpoints which are 2 meters away from the correct trajectory, because we find that viewpoints that are close have very similar visual features, and it would be challenging for the alignment predictor to distinguish them. Off-track but close viewpoints might become noisy data for our contrastive learning model. Furthermore, the agent can actually recover from such points, which makes it unnecessary to differentiate them.

After pretraining, we take the second last layer as the joint representation and concatenate it with the input into the decoder LSTM of the original model. (PREVALENT model in our case.)

3.2.2. Model Architecture

As shown in Figure 4, we keep the original model's vision encoder and language encoder. The output of the language encoder is then fed into a bi-directional LSTM to generate a vector $\widetilde{u}$ representing the instruction. The instruction embedding $\widetilde{u}$ then attends to the 36 vision embeddings of the current viewpoint and generates a attentive vision embedding $\widetilde{f}$.

$$\widetilde{u} = \textbf{LSTM}(u_1, u_2, ..., u_n)$$

$$\widetilde{f} = \sum_{i=0}^{36} a_i * f_i$$

$$a_1, a_2, ... = \textbf{softmax}(A_1, A_2, ...)$$

$$A_i = f_i^T W_{attn} \widetilde{u}$$

We then concatenate $\widetilde{f}$ with $\widetilde{u}$ and feed it to a feed-forward neural network. The last layer is a softmax on two neurons which predicts whether the current viewpoint aligns with the instruction.

3.3. Applying DAGGER on Supervised Learning

Most state-of-the-art work on the R2R task is using the same training scheme e.g. (Hao et al., 2020; Tan et al., 2019; Fried et al., 2018), which is an iterative combination of Imitation Learning and Reinforcement Learning.

In error analysis, we found that a common error in the baseline agent (Hao et al., 2020) is that accumulated mistakes were likely to happen once it steps into an incorrect state and could not get back in future steps. We hypothesize that it is because the agent has never been trained to recover on mistakes since it has never seen this state before in supervised learning.

This motivated our third idea, which is to perform data aggregation in imitation learning (Ross et al., 2011). This process is illustrated in Figure 5, when the model arrives at red nodes, we can compute a recover plan for it (the black arrows).

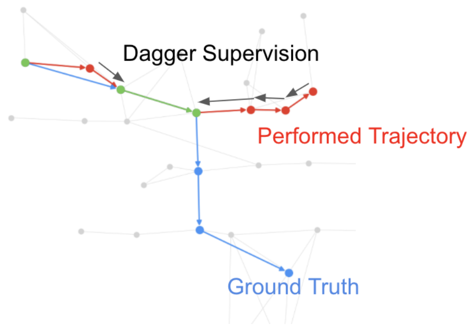


Figure 5. Supervision signals of DAGGER algorithm

R2R dataset is very suitable for DAGGER algorithm, because we can compute the best actions automatically without human labor. To enable the agent to reach off-track viewpoints, we added a round of student-forcing supervised learning with the original teacher-forcing supervised learning, in addition to reinforcement learning. During student-forcing training, the agent takes the action with highest probability at each step, when the viewpoint deviates from the gold path, we compute the shortest get-back path from the current viewpoint to the gold path, and then use the closest viewpoint on the get-back path from the current viewpoint as the current target.

4. Experiment Setup

We begin this section by discussing the main research questions to explore in the experiments for each of our approaches.

External Memory We mainly investigate on how well the agent utilizes the historical trajectory with the help of external memory. Our experiments include a qualitative analysis on repeated error and path lengths. We also look at common failure cases to explore why the agent with external memory doesn't achieve a decent success rate.

Contrastive Learning The main question is whether a specially trained joint representation can improve the agent's success rate in nagivation tasks. We conducted two experiments. The first is the contrastive model's classification accuracy on aligned/misaligned viewpoint-instruction pairs. The second experiment is whether the joint representation learned helps downstream tasks of R2R navigation.

Applying DAGGER in Supervised Learning The core investigation we conduct in this method is whether data aggregation in supervised training improves the agent's performance in navigation and whether it can help guide the agent back on track when it deviates.

4.1. Dataset and Input Modalities

The main dataset we are using in this project is Room-to-Room (R2R). It contains over 4000 routes. Each route is associated with several language instructions describing the same route, and a sequence of view id. Each view id can be mapped to a set of images (see Figure 6 for an example).

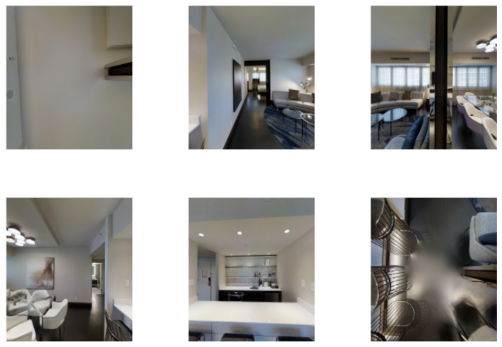


Figure 6. A set of images from different angles and different directions at a single position

Overall, the Room-to-Room (R2R) dataset contains 21,567 navigation instructions with an average length of 29 words. Each instruction describes a trajectory traversing typically multiple rooms.

For the visual modality, most works on R2R utilized the panoramic view representation proposed by (Fried et al., 2018), which enables the agent to "look around" from its current location. The embedding of each image is the concatenation of ResNet-152 features and orientation feature that represents camera heading and elevation poses. And each panoramic view consists of 36 discretized views in 30 degree increments.

4.2. Multi-modal Baseline Model: PREVALENT

To address the problem of limited training data, this work pre-trained on a large amount of image-text-action triplets so that the proposed agent can learn more effectively in this task and generalize better to unseen environments.

4.2.1. Model Architecture

The PREVALENT pre-trained encoder consists of two uni-modal encoders for visual and text modality respectively, and a cross-modal encoder. All encoders are multi-layer transformers (Vaswani et al., 2017). The uni-modal encoder uses the standard self-attention layer, where each position can attend to any position of its own modality in the previous layer, along with the position-wide feed-forward Network (FFN). The cross-modal encoder uses a cross-attention layer to fuse features from both modalities, where each position can attend over all positions in the different modality (see Figure 3 for an illustration of the model architecture).

4.2.2. Pre-training and fine-tuning

Two pre-training tasks were introduced by the authors. For an instruction-trajectory pair $\{X, \tau\}$ from the training dataset, it was assumed that an image-action pair from the trajectory follows an independent identical distribution given the instruction in the pre-training stage: $(f_t, a_t) \overset{iid}{\sim} p(\tau)$.

Attended Mask Language Modeling With each input word x_i randomly replaced with the special token [MASK], the goal is to predict masked words conditioned on the surrounding words $x_{\setminus i}$ and all images f by minimizing the negative log likelihood:

$$L_{\text{MLM}} = -\mathbb{E}_{f\sim p(\tau),(\tau,x)\sim D_E} \log p(x_i|x_{\setminus i}, f)$$

Action Prediction The encoder output at the special token [CLS] contains the fused representation of both modalities. An FC layer is applied on top to predict the action by minimizing the cross-entropy loss:

$$L_{\text{AP}} = -\mathbb{E}_{(a,f)\sim p(\tau),(\tau,X)\sim D_E} \log p(a|x_{[\text{CLS}]}, f)$$

Thus, the agent learns to make correct decisions based on the current image view and the instruction, without referring to the trajectory.

The full pre-training objective is: $L_{\text{Pre-training}} = L_{\text{MLM}} + L_{\text{AP}}$. After pre-training on the shortest-path trajectories from the training environment and synthetic instructions by the Speaker model (6482K image-text-action triplets in total), the model can be fine-tuned for R2R task by feeding the contextualized word embeddings into the Speaker-Follower framework (Fried et al., 2018).

4.3. Experimental Methodology

The R2R dataset consists of four splits: train, validation seen, validation unseen, and test. We evaluate the proposed approaches on the validation seen and unseen data using the following metrics.

Success Rate (SR) Navigation error is the shortest path distance in the navigation graph between the agent's final position (i.e., disregarding heading and elevation) and the goal location. We consider an episode to be a success if the navigation error is less than 3m.

Success rate weighted by path length (SPL) SPL is the ratio between the length of the shortest path and the selected path. SPL penalize the agents that arrive at the final position by randomly walking in the room until reaching the goal.

Coverage weighted by Length Score (CLS) CLS is a more recently proposed (Jain et al., 2019) metric that measures the instruction fidelity of agent path. A higher CLS value indicates that the agent follows the instruction better and it penalizes the agents that reach the goal without following the instruction.

In all of our experiments, the model loads the pre-trained BERT model, and trains other modules from scratch for 20k steps, with the learning rate 1e-4.

5. Results and Discussions

5.1. Overall Model Performance

Table 1 shows that the contrastive learning and DAGGER approaches have achieved similar performance with the baseline agent. On SPL, the agent with contrastive learning has outperformed the baseline in both seen and unseen environments. Also, the agent with DAGGER has outperformed the baseline on SR in seen environments. In terms of generalization to unseen environments, the agent with external memory has experienced the smallest performance drop compared with other agents, although it does not have a superior overall performance.

5.2. External Memory

5.2.1. Qualitative Analysis on Repeated Error

The main idea of external memory is that it remembers which viewpoints the agent has visited and what action the agent has taken. Thus, if the agent reaches a visited viewpoint again, it should avoid taking the same action as before. Otherwise, this situation could easily lead to repeated errors and cyclic behavior.

Specifically, we define a repeated error as the agent revisiting a viewpoint not for the purpose of adjusting its direction or heading (i.e. turning or elevating). Thus, we use a qualitative metric to measure the fraction of repeated mistakes made by the agent among all the navigation waypoints that the agent has visited.

$$\text{Repeated Error Rate} = \frac{\sum_{\text{All trajectories}} \text{Number of repeated errors}}{\sum_{\text{All trajectories}} \text{Number of visited viewpoints}}$$

Table 2 presents the repeated error rates of the baseline agent and our approach in seen and unseen environments. We can see that the agent with external memory has made remarkably fewer repeated errors compared with baseline, especially on validation unseen data, where external memory has reduced the repeated error rate by around 50%.

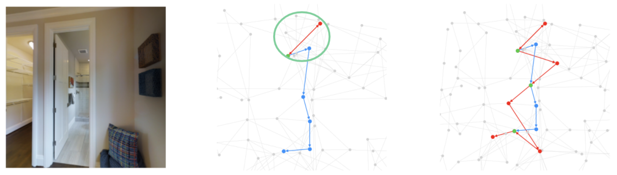


Figure 7. Example of the reduction of repeated error by external memory. Given the instruction "Exit the bedroom. Start down the stairs and stop three steps down", the path taken by the baseline agent (middle); the path taken by the agent with external memory (right).

Figure 7 provides an example of how external memory helps avoid repeated errors. The instruction requires the agent to exit the bedroom, but both agents exited via a different door from expected (left), and didn't find the stairs mentioned in instruction, so they stepped back to the starting viewpoint. At this time, the baseline agent failed to utilize the trajectory information and performed the same action as before, and thus ended up going back and forth between two viewpoints.

	Validation Seen		
	SR	SPL	CLS
PREVALENT(baseline)	0.631	0.580	0.674
External Memory	0.519	0.478	0.619
Contrastive Learning	**0.645**	**0.622**	**0.683**
DAGGER	0.639	0.579	0.674
	Validation UnSeen		
	SR	SPL	CLS
PREVALENT(baseline)	0.549	0.486	**0.593**
External Memory	0.495	0.451	0.585
Contrastive Learning	**0.553**	**0.498**	0.583
DAGGER	0.505	0.414	0.532

Table 1. Performance comparison of our three approaches, using the PREVALENT agent as baseline. All the agents were trained from scratch for 20k iterations with an learning rate of 0.0001. The same pre-trained speaker model and BERT encoder were used.

	Val Seen	Val Unseen
baseline	9.9%	19.9%
External Memory	**7.2%**	**9.9%**

Table 2. Repeated error rates calculated on validation seen and unseen datasets, using the PREVALENT agent as baseline.

While with external memory, the agent could avoid the past mistake, choose a different action to make some real progress.

Another evidence of the repeated error reduction is the decreased path length, as external memory has managed to avoid most situations where the baseline agent would make wasteful repetitive actions. Our hypothesis is that the external memory will play a more important role under some specific situations. Therefore, we create three subsets of the original R2R validation unseen dataset, and examine the agent path lengths.

- Repeated entities sub-dataset contains instructions with at least two repeated nouns, where the agent with external memory should better memorize seen entities and distinguish between the repeated entities.
- Long trajectories sub-dataset is composed of trajectories longer than 6 steps.
- Long instructions sub-dataset picks the top 20% longest instructions.

As shown in table 3, external memory has significantly reduced the path lengths, especially when instructions and trajectories are longer or the same entities are mentioned multiple times.

	Repeated entities	Long trajectories	Long instructions	Overall
baseline	16.3	15.4	17.0	12.6
External Memory	**12.1**	**11.9**	**12.0**	**10.9**

Table 3. Path length comparison of the agent with external memory and the baseline agent on three sub-datasets.

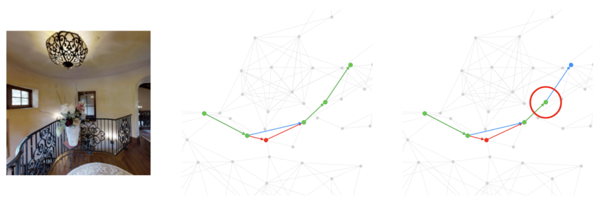


Figure 8. Example of the early stop error. Given the instruction "Stop ... just before the stairs", the path taken by the baseline agent (middle); the path taken by the agent with external memory (right).

5.2.2. Failure Cases

Although the path length is significantly reduced, we also observe some common failure cases, causing the relatively low success rates. As shown in Figure 8, the instruction tells the agent to stop by the stairs. The agent successfully goes to the stairs, but our model stops when it merely sees the stairs. This is a common error case with external memory. We find many times that the agent stops on the correct trajectory but 1 or 2 steps before it reaches the actual goal. Our hypothesis is that the agent will stop when it does not find action candidates which are confident enough, which is worth more attention in the future.

5.3. Contrastive Learning

In order for the joint representation to help the R2R navigation task, two steps must be successful. Firstly, the joint representation learnt must be meaningful, which in our case requires that the representation must be able to predict whether a viewpoint-instruction pair is aligned. Secondly, the joint representation must contain information helpful to navigation tasks. We present our experiment results on both steps mentioned above.

5.3.1. Alignment-Aware joint representation

	Positive	Negative	Total
Val Seen	89.4%	64.8%	77.1%
Val Unseen	83.5%	63.3%	73.3%

Table 4. Contrastive model's classification accuracy on viewpoint-instruction pairs. The model is supposed to predict 1 if the viewpoint is from the correct trajectory according the instruction.

Table 4 shows the classification accuracy of the contrastive model on viewpoint-instruction pairs. Under both seen and unseen environment, the accuracy is over 70%. We further plot a t-sne graph for the joint embeddings of correct and wrong viewpoint-instruction pairs(Figure 9), and the clustering of two groups is obvious.

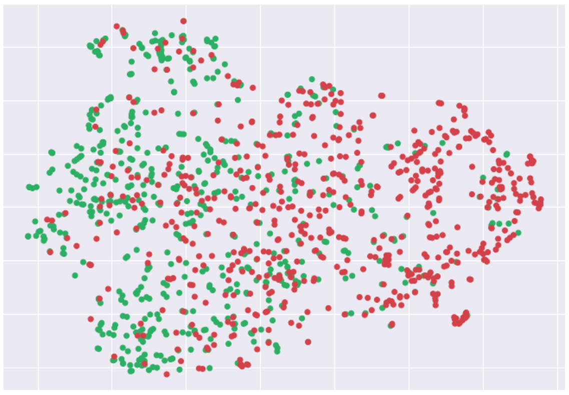


Figure 9. T-sne plot of joint representations of aligned and misaligned viewpoint-instruction pairs. The green points are positive pairs, and the red points are negative pairs.

Both the alignment classification task and the t-sne graph show that the learned joint representation contains alignment information of the current viewpoint.

However, the individual results for positive pairs and negative pairs (Table 4) indicate that the model is prone to make positive predictions. We think this is because objects of the same type, for example, a table, can appear at multiple places in the house. Without trajectory information, the classification model is unable to tell the difference and outputs true for all.

5.3.2. Joint Representation for navigation tasks

It is shown in Table 1 that the agent pretrained with contrastive learning outperformed the baseline model in almost all metrics under both validation seen and validation unseen environments. Since the agent used in this method is the same as the baseline model except for the additional joint representation, the improvement on success rate is a strong indicator that the joint representation is helpful in the R2R navigation task.

To further prove that the joint representation is useful in navigation tasks. We calculate the cosine similarity between the joint embedding of the current viewpoint and the next available viewpoints during navigation (Table 5). The current viewpoint has a higher cosine similarity with the next correct viewpoint compared to other wrong moves. This partially indicates that the agent can simply predict the next move by moving to viewpoints with similar joint embeddings.

	Correct Action	Wrong Action
cosine sim	0.54	0.50

Table 5. Average cosine similarity between current viewpoints and the viewpoints after taking correct actions or wrong actions.

We further tested the joint representation on another agent EnvDrop (Tan et al., 2019) (Table 6). The EnvDrop agent with joint representation can achieve the same success rate with only 25% of training iterations. And the learning curve showed that it has great potential in increasing the success rate with more iterations. Therefore we believe the contrastively learnt joint representation can be beneficial to all late-fusion based models.

	Original	With joint representation
# iters	$80k$	$17.7k$

Table 6. Another model, EnvDrop's performance with/without the joint representation. Due to limited time, we only show number of iterations achieving the original model's final success rate of 46.0%.

5.4. Applying DAGGER on Supervised Learning

Agent trained with data aggregation during supervised learning outperforms the baseline agent in the validation seen dataset, as shown in Table 1. The core idea of using DAGGER is to supply the off-track viewpoints with the correct target as part of the training data in supervised learning phase. So, to evaluate the effect in getting-back, we define the get-back rate as the chance of getting back on-track when the agent deviates, and computed the get-back rate of the model trained with DAGGER and the baseline model without DAGGER.

$$\text{Get Back Rate} = \frac{\sum_{\text{All Trajectories}} \text{Number of times agent gets back on gold path}}{\sum_{\text{All Trajectories}} \text{Number of times agent deviates from gold path}}$$

As shown in Table 7, the model train with DAGGER has a much higher get-back rate, indicating that the agent has gained the knowledge of how to get back to the gold path when it finds itself not on-track.

	VAL SEEN	VAL UNSEEN
WITHOUT DAGGER	44.98%	53.86%
WITH DAGGER	**51.29%**	**69.02%**

Table 7. Get-back rate when the agent is off-track comparison of the agent trained with and without DAGGER.

Observing the higher get-back rate of the agent trained with DAGGER, and it's lower success rate compared with the baseline agent in the unseen validation dataset, we further measured the cause of this by checking which viewpoint on the gold path the agent is more likely to return to. The back-to-previous-state rate measures whether the returned point has already been visited by the agent, meaning the agent is essentially doing a reversed action to get back.

	VAL SEEN	VAL UNSEEN
WITHOUT DAGGER	37.63%	44.98%
WITH DAGGER	**44.51%**	**67.63%**

Table 8. Back-to-previous-state rate comparison of the agent trained with and without DAGGER.

From Table 8, we conclude that the reason for lower success rate even with higher get-back rate is that the agent gets back to visited viewpoints on the gold path, instead of moving forward.

In addition to the statistical results on the agent performance, we have also observed an interesting finding during training. The agent with DAGGER improves performance faster in the earlier stage of training, but then the baseline agent can catch up with more iterations of training. Since the agent is trained with iterative supervised learning and reinforcement learning, we believe in the R2R task, training with reinforcement learning where an agent is allowed to explore in the environment, the agent can also learn how to get back on track, just at a slower rate compared with using DAGGER and train the agent in a supervised manner.

6. Conclusion

We examined three approaches targeted at different problems, 1) using external memory to form a robust trajectory representation and reduce repetitive wrong moves, 2) adopting contrastive learning to learn a alignment-aware visual-language joint embedding and improve trajectory finding, 3) applying DAGGER method to teach the agent correct actions when deviated. We observed strong performance improvement brought by contrastive learning under all environments and the DAGGER method under the *val_seen* environment. And even though external memory did not beat the baseline model in success rate, it clearly showed advantages in reducing path length and avoiding repetitive wrong moves. In summary, all three methods solved their targeted problem but more work might be needed to refine the external memory's architecture to improve its performance on success rate.

7. Future Directions

External Memory There are several potential refinements for the external memory that could be done in the future, which are promising in achieving better performance.

- Multi-head Attention. Currently, for both the write and read operation, we are using a uni-head attention. We think it is worth trying replacing it with multi-head attention.
- Position Encoding. Attention model treats all the keys and values equally, ignoring their position information. Therefore, adding a position encoding to the memory slots should also be tried.
- Predicting Stop. As shown in the experiment results, the model with external memory is suffering from stopping before the goal has reached. We suggest to train a separate stop prediction module in the future, which might help alleviate the problem of early stop.

Contrastive Learning We can apply the contrastively learnt joint representation to other late-fusion based navigation models and compare the effectiveness of the joint representation.

DAGGER Observing the faster learning rate using DAGGER and the advantage of exploration in reinforcement learning, one potential experiment that can be done with the agent trained using DAGGER is to change the training scheme by putting more iterations of supervised learning with DAGGER in the earlier stage and allows for more reinforcement learning rollouts in the later stage instead of constant iterations between supervised learning and reinforcement learning.

References

Anderson, P., Wu, Q., Teney, D., Bruce, J., Johnson, M., Sünderhauf, N., Reid, I., Gould, S., and van den Hengel, A. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3674–3683, 2018.

Fried, D., Hu, R., Cirik, V., Rohrbach, A., Andreas, J., Morency, L.-P., Berg-Kirkpatrick, T., Saenko, K., Klein, D., and Darrell, T. Speaker-follower models for vision-and-language navigation. In *Advances in Neural Information Processing Systems*, pp. 3314–3325, 2018.

Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwińska, A., Colmenarejo, S. G., Grefenstette, E., Ramalho, T., Agapiou, J., et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476, 2016.

Gruber, M., MacMillan, I. C., and Thompson, J. D. Look before you leap: Market opportunity identification in emerging technology firms. *Management science*, 54(9): 1652–1665, 2008.

Hao, W., Li, C., Li, X., Carin, L., and Gao, J. Towards learning a generic agent for vision-and-language navigation via pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13137–13146, 2020.

Hill, F., Tieleman, O., von Glehn, T., Wong, N., Merzic, H., and Clark, S. Grounded language learning fast and slow. *arXiv preprint arXiv:2009.01719*, 2020.

Huang, H., Jain, V., Mehta, H., Ku, A., Magalhaes, G., Baldridge, J., and Ie, E. Transferable representation learning in vision-and-language navigation. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 7404–7413, 2019.

Jain, V., Magalhaes, G., Ku, A., Vaswani, A., Ie, E., and Baldridge, J. Stay on the path: Instruction fidelity in vision-and-language navigation. *arXiv preprint arXiv:1905.12255*, 2019.

Ma, C.-Y., Wu, Z., AlRegib, G., Xiong, C., and Kira, Z. The regretful agent: Heuristic-aided navigation through progress estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6732–6740, 2019.

Majumdar, A., Shrivastava, A., Lee, S., Anderson, P., Parikh, D., and Batra, D. Improving vision-and-language navigation with image-text pairs from the web. *arXiv preprint arXiv:2004.14973*, 2020.

Ross, S., Gordon, G., and Bagnell, D. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635, 2011.

Tan, H., Yu, L., and Bansal, M. Learning to navigate unseen environments: Back translation with environmental dropout. *arXiv preprint arXiv:1904.04195*, 2019.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Wang, X., Huang, Q., Celikyilmaz, A., Gao, J., Shen, D., Wang, Y.-F., Wang, W. Y., and Zhang, L. Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6629–6638, 2019.