SeqVista: Sequence to Sequence Model for Vision-Language Navigation

Z. H. Akgül, A. Bakhshayesh, H. Huzaifa, E. Karataş, F. Khan

Bilkent University, Ankara, 2023

Abstract

The SeqVista project represents a significant undertaking within the field of Artificial Intelligence, with a particular emphasis on advancing Vision-Language Navigation (VLN). The core objective is to develop a robust Sequence-to-Sequence (Seq2Seq) model capable of training an autonomous agent for effective VLN. This requires equipping the agent with the proficiency to interpret and execute natural language instructions within unseen environments by integrating advanced natural language processing and computer vision techniques. A key aspect of this project involves taking advantage of the Matterport 3D simulator in conjunction with the Room-to-Room (R2R) dataset, which provides a graph-based environment rich in real-world imagery and diverse instruction-trajectory pairs.

Keywords—*Artificial Intelligence, Vision-Language Navigation, Sequence-to-Sequence Model, Natural Language Processing, Computer Vision, Matterport 3D Simulator, Room-to-Room Dataset, LSTM, RESNET-152, Autonomous Agents.*

I. Introduction

SeqVista is an attempt to explore the realm of VLN within Artificial Intelligence. With our understanding of previously available resources, we aim to craft a Seq2Seq model that can train agents proficient in interpreting natural language instructions to achieve tasks in the Matterport3D Simulator and beyond. The main aim of the project is to study the effect of removing the attention component in a Seq2Seq model. At the time of writing this report, most of the coding parts were done, as well as setting up the simulator, obtaining and processing the datasets, and more. The goal is to further train the agents and display the obtained trajectories to view how the agent works in real time.

II. Literature

A detailed literature review is quite important to help understand what is already known in the area of research. By identifying gaps, the literature review reasons the research. Additionally, gathering together the key ideas from existing studies helps to build a strong foundation. Learning from past studies allows us to make smart choices in the project's design and methodology.

This part is divided into five subsections. First, key techniques and will be discussed. Then, a comparative analysis will be made, discussing the future directions.

A. Vision-Language Navigation

Research on natural language command of robots in unstructured environments has been ongoing for decades. Previous approaches often simplified the problem by either pre-defining navigation goals and objects or operating in limited perception environments [5]. The development of new benchmark datasets has driven advances in image captioning, visual question answering, and visual dialog. These have enabled end-to-end training on raw image data but have not allowed for agent movement or camera control, a gap the R2R benchmark aims to fill [6, 7, 8, 9].

Existing Approaches in Vision-Language Navigation

- *Natural Language Processing (NLP) Techniques:*

  Natural language processing techniques are commonly used in order to understand the navigation instructions, as seen in papers. For example, Mei, Bansal, and Walter use a neural mapping approach to map instructions to a sequence of actions. Their purpose is to train a neural network to be able to predict the sequence of actions which an agent is supposed to take based on the input instruction and visual features of the environment. The network is trained on a large dataset of the pairs of instruction-action, and the network learns to generalize to new instructions and environments [10]. Similarly, Tellex et al. [11] and Fasola et al. [12] focus on mapping verbs and spatial relations of verbs to certain actions in the environment. For example, the verb "go" might be associated with the action of moving forward, while the spatial relation "left of" might be associated with the action of turning left. Agents can navigate based on natural language instructions by mapping these natural language instructions into actions in the real world like environment. One other research, proposed by [10], introduces an end-to-end, sequence-to-sequence approach to mapping natural language instructions to actions where the local and the observable world state is given, using a bidirectional LSTM-RNN model with a multi-level aligner. However, it is also mentioned that free-form instructions in unknown environments are problematic due to their ambiguity and complexity, which is a known issue in NLP techniques. To deal with the problem, the paper proposed a model that uses a multi-level aligner to focus on certain parts of a sentence that are relevant to the current world state. The paper then analyzes its performance through a series of decomposition experiments. Overall, NLP techniques are used to generate a sequence of actions that an agent can take to navigate in the environment based on natural language instructions, even in complex and ambiguous environments.
- *Computer Vision Techniques:*

  In VLN, several computer vision techniques are commonly used to make agents analyze and respond to natural language instructions in a visual environment. For example, CNN is used for feature extraction to understand the content of the environment for decision-making. For example, in [13], pre-trained CNN Resnet-152 on ImageNet to extract feature vectors of images implemented. Other than CNN, attention mechanisms are also used in studies. In [2], attention mechanisms are used to focus on relevant parts of an image, allowing the model to concentrate on the specific details in the natural language instructions. Several obstacles occur with these techniques. For example, one of the problems is the generalization to previously unseen environments. In [14], it is mentioned that the Multimodal Indoor Simulator for Navigation in Complex Environments (MINOS) evaluates the agents in environments that they were not trained in, allowing for the study of generalization. This is important for VLN systems to be able to navigate effectively in new and unfamiliar environments.

- *Integration of NLP and Computer Vision:*

Combining language understanding and visual perception can cause several challenges, especially if the agent only has access to visual content without the ability to interact with the environment. To deal with the issue, in [15], Interactive Question Answering (IQA), which requires the agent to interact with a dynamic environment, is proposed. IQA involves tasks such as navigation, acquiring an understanding of the environment, interacting with objects, and planning and executing actions based on questions asked. Furthermore, Multimodal Attention Mechanisms are also used to operate across both language and vision modalities. In [2], attention mechanisms that function across both language and vision domains enhance the model's ability to concentrate on relevant elements within textual instructions and visual environments, improving navigation precision.

Benchmark Datasets for VLN

Benchmark datasets are quite significant in the development of VLN systems. The R2R benchmark is one of those, providing a diverse set of instruction-trajectory pairs associated with Matterport 3D environments. This dataset, when associated with Matterport 3D, offers a robust testing environment for VLN agents, allowing evaluations on realistic, building-scale environments. The availability of such benchmark datasets is significant for training models and assessing their performance in unseen scenarios in VLN research. Similar to research using R2R Benchmark, as in [2], this dataset is also used in SeqVista because of its ability to be associated with the Matterport 3D Simulator.

B. Sequence-to-sequence Models

Agents in various papers are implemented by the Sequence-to-Sequence (Seq-2-seq) model. Seq-2-Seq models are recurrent neural network (RNN) models based on LSTM-based architecture [16]. It is worth mentioning that most of the papers implement the Seq-2-Seq model with an attention mechanism [16]. However, the Seqvista project implemented the Seq-2-Seq model **without an attention mechanism** to obverse the importance of the attention mechanism in the related Seq-2-Seq models.

Detailed information related to the attention mechanism should be mentioned to highlight the difference between SeqVista and other papers implementing the Seq-2-Seq model.

An attention mechanism functions as a crucial component within a neural network framework[17]. It operates by determining the significance of various parts of the source data at each step of the decoder's process. This mechanism allows for a more nuanced approach compared to traditional methods where the encoder had to encapsulate the entire source information into a single, compact vector. Instead, it provides detailed representations for each token in the source - for instance, it can offer representations for every state in a Recurrent Neural Network (RNN) rather than just the final state[17]. This enables the model to focus selectively on different parts of the input data, enhancing its ability to understand and process complex patterns and dependencies, especially in tasks such as language translation or image recognition where context and specific details are crucial[17].

Seq Models in Navigation

Matterport 3D simulator action space are state-dependent; agents are allowed to make pre-defined actions based on choices[13]. Since the Matterport 3D camera takes visuals every 30 degrees, the model and navigation of Seq-2-Seq models are defined to move every 30 degrees.

For image observation, the papers that have been read implement pre-trained CNN Resnet-152 on ImageNet to extract feature vectors of images [13]. That is, an embedding is learned from every action in the action space and then takes relative actions in the navigation.

Most of the papers predict future actions based on the attention mechanism explained in detail above. It computes an attentional hidden state and calculates the predictive distribution over the next states as softmax[18]. Although it is an enormous advantage to increase the model's performance, Seqvista is based on the Seq-2-Seq model without an attention mechanism to highlight the importance of an attention mechanism in these models.

C. Challenges and Open Problems

While VLN has developed significantly in recent years, several challenges persist in making these systems robust and adaptable. One crucial problem is the generalization of models to previously unseen environments. Addressing this challenge is critical for VLN systems to navigate effectively in new and unfamiliar settings [14]. This issue also appears in SeqVista but is not analyzed further to avoid misdirecting the project scope. Another significant challenge involves integrating natural language processing (NLP) and computer vision, particularly when the agent has limited interaction with the environment [15]. Interactive Question Answering tasks, as proposed in [15], show the complexity of tasks requiring dynamic interactions in VLN systems. Overcoming these challenges will contribute to developing more versatile and capable VLN agents.

D. Comparative Analysis

A comparative analysis of existing approaches in Vision-Language Navigation with SeqVista allows for the analysis of diverse techniques. Natural Language Processing techniques, as studied by Mei, Bansal, Walter, Tellex, et al., and Fasola and Mataric, focus on mapping the natural language instructions to sequences of actions, grounding verbs, and spatial relations to actions in the environment [10, 11, 12]. SeqVista also uses NLP techniques to map the instructions to specific actions like these studies. Computer Vision techniques, such as those discussed in [14], emphasize the evaluation of agents in previously unseen environments. Integrating NLP and Computer Vision introduces challenges associated with dynamic environment interaction. SeqVista, with the advantage of Matterport 3D simulator, experiences a dynamic environment interaction, differentiating from studies such as [14]. The Seq-2-Seq model, a recurrent neural network architecture, is widely adopted [16], with attention mechanisms enhancing model performance in various papers [17]. However, SeqVista intentionally excludes an attention mechanism to investigate its impact on Seq-2-Seq models in the VLN context. This comparative analysis guides the SeqVista project in understanding the strengths and limitations of different methodologies.

E. Conclusion and Future Directions

In conclusion, the literature review provides valuable insights into the existing state of Vision-Language Navigation. Essential techniques have been explored, including NLP and Computer Vision integration, Seq-2-Seq models, and attention mechanisms. It has been observed that the lack of attention mechanisms reduces the agent's overall performance, highlighting the importance of such mechanisms. The challenges, such as generalization to unseen environments and dynamic environment interaction, show the complexity of VLN systems. The SeqVista project aims to contribute to this massive field by intentionally excluding an attention mechanism in the Seq-2-Seq model, emphasizing the importance of this component in VLN models. Future directions include
Further experimentation without attention mechanisms.

- Training the Model with attention mechanism using the code by [2] for fair comparison of the model's performance with & without attention mechanism
- Creating an Evaluation setup for the Seq-Seq agent for its evaluation on unseen test split of dataset
- Addressing challenges in generalization.
- Exploring more sophisticated interactive tasks within the VLN domain.

III. Description

The project aims to develop a Sequence-Sequence (Seq) model for training an autonomous vision-language navigation (VLN) agent. VLN refers to the navigation of embodied agents in an unseen environment based on natural language instructions. The agent must understand the natural language instructions using the natural language processing and then use computer vision to identify potential landmarks mentioned in the instructions to initiate its navigation actions. As an example, the agent might be required to execute the instruction, *"Walk down the stairs to the bottom of the staircase. Continue down the next small flight of stairs towards the bathroom at the lower level"*.

A. Simulator

For the implementation of the VLN agent, the Matterport 3D simulator was used. The Matterport 3D simulator is a graph-based environment based on the Matterport 3D dataset [1], which consists of 90 building-scale environments. The Matterport 3D simulator was used in conjunction with the Room-to-Room (R2R) dataset [2]. This dataset consists of a large number of diverse instruction-trajectory pairs associated with Matterport 3D environments. The action space in this simulator consists of the following actions:

(1) Forward: This action moves the agent to the navigable node, which is closest to the center of the agent's field of view (FOV).

(2) Right, Left: These actions would change the camera heading by 30 degrees.

(3) Up, Down: These actions would change the camera elevation by 30 degrees.

(4) Stop: This special action is initiated for the termination of the navigation episode.

At each time step, the output of the Matterport 3D simulator consists of:

- First-person RGB observation of the environment.
- The set of next navigable nodes from each node.

B. Seq-to-Seq Model

The VLN problem can be solved using a Sequence-Sequence model, also called the encoder-decoder model, which takes in a sequence of natural language instructions to produce the sequence of navigational actions, as shown in Fig 1. The input to the encoder-decoder model is the sequential instruction $w = \{w_1, w_2, w_3 ... w_n\}$, where w_i represents a learned embedding vector for each of the n-word tokens present in the instruction. Similarly, the image features for RGB observations were also computed beforehand using RESNET-152 architecture, which is pre-trained on ImageNet architecture and makes use of convolutional neural networks for the extraction of image features. We represent the extracted image features by f_t. Also, the embeddings for each of the actions were learnt separately. The encoder LSTM takes in the instruction sequence w to compute the final hidden state h_n from the encoder context $h = \{h_1, h_2, h_3 ... h_n\}$ as:

$$h_n = LSTM_{enc}\{w_n, h_{n-1}\}$$

This final hidden state of encoder LSTM is passed to the decoder LSTM for **outputting** the sequence of navigational actions. At each time step t, the decoder observes the hidden state h_n, the embeddings of the previous action, a_{t-1} and the representation f_t of current RGB observation o_t to compute the raw probabilities for the next action. The action embeddings and the image representations are concatenated into a single vector q_t. **So, the decoder LSTM operates as:**

$$h_t' = LSTM_{dec}\{q_t, h'_{t-1}\}$$

where h_t' gives us the raw probabilities for next action. These raw probabilities are passed through the softmax function to obtain the final output in terms of navigational action in the simulation environment as

$$a_t = softmax(h_t')$$

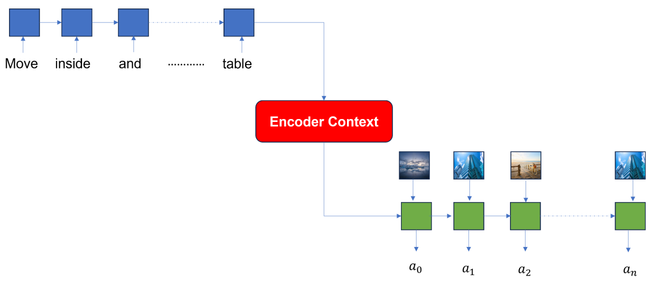


Figure 1: Seq-Seq Model

IV. Progress & Milestones Achieved

This section is spared for a detailed explanation of the work done until this report.

A. *Simulator*

In the pursuit of comprehending the MatterSim simulator and its API, significant milestones were achieved through systematic exploration and practical application. This section outlines the milestones reached during the understanding of the simulator, emphasizing a hands-on approach.

Technical Exploration:

- **Configuration Mastery:** The initial phase involved mastering the configuration parameters of the MatterSim simulator. Functions like setCameraResolution, setPreloadingEnabled, setDepthEnabled, setBatchSize, and setCacheSize were dissected for their role in the simulator setup.

- **Simulator Initialization:** The simulator was initialized using the initialize function, setting the stage for subsequent interactions. A detailed understanding of parameters like scan IDs, viewpoint IDs, headings, and camera elevations was crucial for accurately defining the simulator's starting state.

- **Action Execution:** The core of agent interaction was explored through the makeAction function. The program executed forward movements with varying heading and elevation changes. This hands-on experience facilitated a deep understanding of how the agent navigates within the simulator.

- **State Inspection:** Regular checks of the simulator state using the getState function provided insights into the dynamic changes during and after agent actions. Extracting information such as scanId, step count, RGB and depth images, location details, and heading facilitated a comprehensive understanding of the simulator's inner workings.

Programming Accomplishments:

- **Code Implementation:** A simple Python program was developed to showcase the practical application of the acquired knowledge. The program executed a sequence of actions, moving the agent forward in different elevations and headings for a specified number of steps(see Appendix 1 for the code).

- **Iterative Adjustment:** The program's iterative nature allowed for continuous adjustments based on real-time observations of simulator behavior. This iterative process contributed to a more nuanced understanding of the simulator's response to different inputs.

B. *R2R Cleanup*

For the development of the SeqVista project, an open-source Matterport 3D Simulator Dataset is utilized [1]. Since the key aspect of this project relies on the graph-based Matterport 3D Simulator, the R2R Dataset is employed in conjunction with the Matterport 3D Simulator.

Initially, the original Matterport 3D R2R Dataset consisted of 90 houses composed of 1.3 TB of data [3].
Due to hardware (especially RAM, GPU & storage capacity) and time constraints of this project, it is decided to use 10 randomly selected houses, which are partitioned into 3 different categories:

- **Training:** 7 Houses - 70% of data
- **Testing:** 2 Houses - 20% of data
- **Validation (Unseen):** 1 House - 10% of data

Randomly selected 10 houses composed of 150 GB of data, which is decided to be enough in order to get meaningful results, taking account of limitations mentioned above favoring optimality.

Houses in the Matterport 3D R2R Dataset are labeled with their scan codes. The dataset consists of 4 different JSON files, which are named as follows:

- **R2R_test.json**
- **R2R_train.json**
- **R2R_val_seen.json**
- **R2R_val_unseen.json**

These JSON files give insight about the scan (house), the path ids' related to these scans, their headings (in radians) and instructions (in natural English language) to accomplish these moves.

The aforementioned randomly selected houses' scan IDs for this project are as follows:

- 17DRP5sb8fy
- 1LXtFkjw3qL
- 1pXnuDYAj8r
- 29hnd4uzFmX
- 2azQ1b91cZZ
- 2n8kARJN3HM
- 2t7WUuJeko7
- 5LpN3gDmAk7
- 5q7pvUzZiYa
- 5ZKStnWn8Zo

To filter selected houses, R2R_data_cleaning.py script is developed. This script basically filters related houses depending on their scan IDs and returns to JSON files named as follows:

- **R2R_test_processed.json**
- **R2R_train_processed.json**
- **R2R_val_seen_processed.json**
- **R2R_val_unseen_processed.json**

Then, mentioned processed JSON files are used in the training of model.

C. Seq-Seq Agent Development & Training

For the development of the Seq-Seq agent using LSTMs, an open-source pytorch library v1.5.0 was used. The sentences were tokenized in white spaces. We follow the standard hyperparameter values used by [2]. The hidden size of neural network layers was set to 512. The size of action and word embeddings was set to 32 and 256, respectively. The image features were extracted from the second-last layer of RESNET-152 architecture, which has a size of 2048. We set the maximum episode length to 20. The loss criterion used in this case was a cross-entropy loss. We use the Adam optimizer for the optimization of encoder and decoder parameters. We use a learning rate of 0.0001 and a weight decay of 0.0005 in order to train our agent. The dropout ratio to avoid overfitting is set to 0.5. The agent was trained using two different feedback approaches, namely "sample" and "teacher" feedback. In sample feedback, the agent uses the predicted previous action a_t based on the learned model for getting the probability distribution over the next action in the output stage of the encoder-decoder model. However, in the "teacher" feedback approach, the agent always uses the correct ground truth in the previous action a_t* instead of the predicted previous action to get the probability distribution for the next action. The agent was trained on a core i9 machine with 125 GB RAM and incorporating an NVIDIA GeForce RTX 3090 Ti/PCIe/SSE2 GPU. We trained the agent on a batch size of 100 with a fixed number of iterations, which was 20000 for the "sample" feedback approach and 5000 for the "teacher" feedback approach. It took approximately 3 hours for training the agent using "sample" feedback approach, while "teacher" feedback approach took about 45 minutes. The model weights were saved in a separate directory to be used for evaluation on the test split of the R2R dataset. The training & validation loss curves along with the success rate are shown in Fig 2.

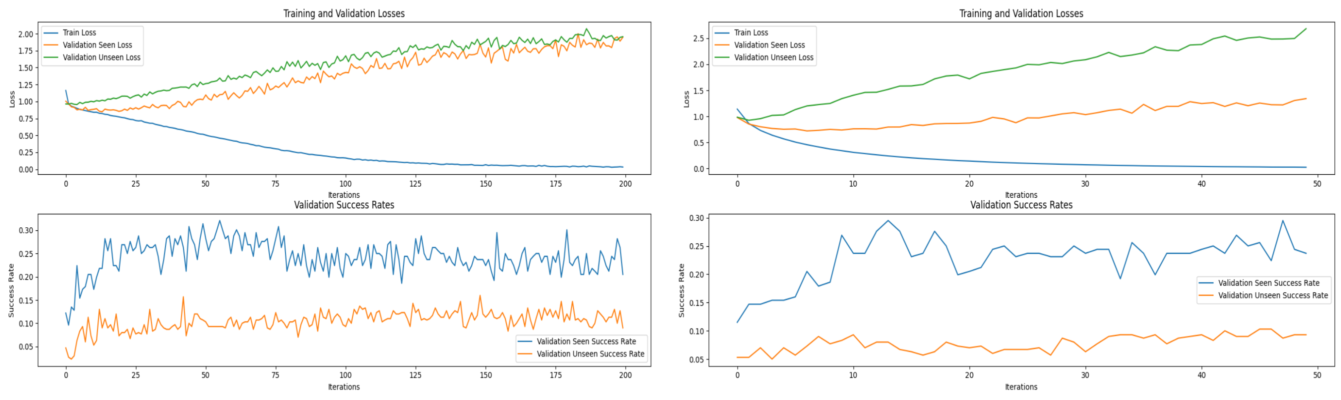


Figure 2: Training, Validation Losses & Success Rate for Sample (Left) and Teacher Feedback (Right)

D. Results

From Fig 2, we can see that the model converges quickly when using the teacher feedback approach as compared to the sample feedback approach. However, the success rate for teacher feedback is slightly lower than that of sample feedback in our case. It is interesting to note that the reverse is true when we use the attention mechanism along with the Seq-Seq model (Table 1). We leave the investigation of this research question as a future task.

The results from Table 1 show that there is a huge drop in success rate for both validation seen and unseen dataset splits in our case, where we have removed the attention component of the Seq-Seq model as compared to the Seq-Seq model with attention developed by [2]. The attention mechanism is crucial in neural networks as it allows the model to determine the significance of different parts of the input data at each step of the decoding process. Unlike traditional methods that compress all source information into a single vector, attention provides detailed representations for each token, enabling the model to selectively focus on different parts of the input. So, our experiments reinforce this idea that the attention mechanism is crucial in the cross-modal grounding of RGB observations at each time step on certain parts of input natural language instructions.

The loss of performance could also be the result of the significantly small size of the dataset, with 10 houses, as compared to 90 houses in the case of [2]. For this purpose, we aim to run the implementation of the Seq-Seq agent with attention on 10 houses using the code by [2] for fair comparison as a future task.

Training Approach	**Val Seen Success Rate**	**Val Unseen Success Rate**
Sample Feedback [Anderson et al.]	38.6%	21.8%
Teacher Feedback [Anderson et al.]	27.1%	19.6%
Sample Feedback (Ours)	**20.5%**	**9.0%**
Teacher Feedback (Ours)	**23.7%**	**9.3%**

Table 1: Results of Training

E. Javascript Trajectory Visualization

To be able to properly visualize, view, and save the agent trajectories, an HTML and Javascript application is created. This application will allow us to view the agent's trajectories in a first-person point-of-view and also download them as videos [4]. With this application, you can choose a trajectory file, play it, download it, set the camera parameters, and change the index to view different trajectories from the selected file.

F. Detailed Literature Review

In-depth literature review parallel to the model development with a strategic approach was conducted. Initially, key headings guiding the search for relevant conference papers, research studies, and related publications are identified. The techniques used and the challenges faced in the studies' are examined carefully and compared with how our project, SeqVista, is set up. This analysis helped to see the specific challenges we're dealing with, guiding us to make well-informed decisions. Finally, open problems are discussed for future studies in this area.

V. Timeline

Up to date, when this report was written, most of the coding part of the project has been done. We successfully completed extensive data cleaning on the raw dataset, carefully set the Matterport 3D simulator, and ran the simulator based on processed data. We obtained our initial results, which are explained in detail in the aforementioned sections.

We accomplished more than what we have proposed in the timeline of the final proposal report, which is above 70% of the project completion status. Our goal for the rest of the 3 weeks till the final report and presentations is to setup an evaluation environment for evaluating the agent on the test split of the dataset, develop our agents further in the model to get better evaluation results and obtain the trajectory visualization for generated trajectories in the form of video. We also aim to run the implementation of the Seq-Seq agent with attention to our dataset, which contains 10 houses. We would be using the code by [2] so that we get a fair comparison in results where we have an equal size of the dataset for both cases.

VI. Work Distribution

- **Zeynep Hanife Akgül:** Literature Review, Challenges, and open problems, Comparative analysis, Matterport 3D Simulator set up trial run using AWS Linux instance.
- **Arshia Bakhshayesh:** Matterport 3D R2R Dataset downloading, Matterport3D simulator installation, Matterport3D API configuration and mastery
- **Huzaifa Huzaifa:** Development of Seq-Seq model, Training & Evaluation of Learnt Agent
- **Emre Karataş:** Matterport 3D R2R Dataset downloading & preprocessing, Literature Review, Code Debugging Seq-2-Seq Model.
- **Faaiz Khan:** Javascript Trajectory Visualization.

References

[1] Matterport3D. "Learning from RGB-D Data in Indoor Environments." Accessed Oct. 23, 2023. [Online]. Available: https://niessner.github.io/Matterport/.

[2] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, and A. V. D. Hengel, "Vision-and-Language Navigation: Interpreting Visually-Grounded Navigation Instructions in Real Environments," ArXiv, 2017. [Online]. Available: https://arxiv.org/abs/1711.07280.

[3] "Matterport3D Scans." Accessed at: https://kaldir.vc.in.tum.de/matterport/v1/scans.txt.

[4] P. Anderson, "MatterPort3D Simulator," Github Repository, 2021. [Online]. Available: https://github.com/peteanderson80/Matterport3DSimulator.

[5] T. Winograd, "Procedures as a Representation for Data in a Computer Program for Understanding Natural Language," Technical Report, Massachusetts Institute of Technology, 1971.

[6] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh, "VQA: Visual Question Answering," in ICCV, 2015, pp. 2, 4, 5.

[7] X. Chen, T.-Y. L. Hao Fang, R. Vedantam, S. Gupta, P. Dollar, and C. L. Zitnick, "Microsoft COCO Captions: Data Collection and Evaluation Server," arXiv preprint arXiv:1504.00325, 2015, pp. 2, 4.

[8] A. Das, S. Kottur, K. Gupta, A. Singh, D. Yadav, J. M. F. Moura, D. Parikh, and D. Batra, "Visual Dialog," in CVPR, 2017, pp. 2.

[9] Y. Goyal, T. Khot, D. Summers-Stay, D. Batra, and D. Parikh, "Making the V in VQA Matter: Elevating the Role of Image Understanding in Visual Question Answering," in CVPR, 2017, pp. 2.

[10] H. Mei, M. Bansal, and M. R. Walter, "Listen, Attend, and Walk: Neural Mapping of Navigational Instructions to Action Sequences," arXiv preprint arXiv:1506.04089, 2015.

[11] S. A. Tellex, T. F. Kollar, S. R. Dickerson, M. R. Walter, A. Banerjee, S. Teller, and N. Roy, "Understanding Natural Language Commands for Robotic Navigation and Mobile Manipulation," 2011.

[12] J. Fasola and M. J. Mataric, "Using Semantic Fields to Model Dynamic Spatial Relations in a Robot Architecture for Natural Language Instruction of Service Robots," in Intelligent Robots and Systems (IROS), 2013, pp. 143–150.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in CVPR, 2016, pp. 6.

[14] M. Savva, A. X. Chang, A. Dosovitskiy, T. Funkhouser, and V. Koltun, "MINOS: Multimodal Indoor Simulator for Navigation in Complex Environments," arXiv:1712.03931, 2017.

[15] D. Gordon, A. Kembhavi, M. Rastegari, J. Redmon, D. Fox, and A. Farhadi, "IQA: Visual Question Answering in Interactive Environments," in CVPR, 2018.

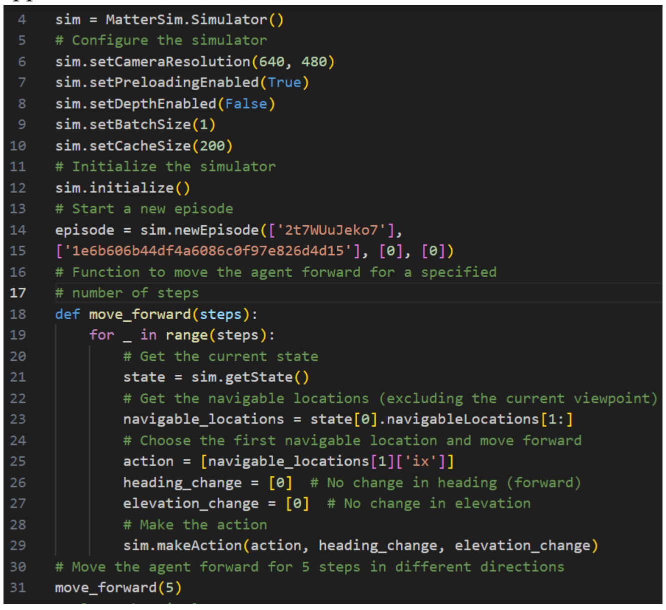
[16] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," Neural Computation, 1997, pp. 6.

[17] "Seq2Seq and Attention," Lena Voita's NLP Course, Accessed at: https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html.

[18] M.-T. Luong, H. Pham, and C. D. Manning, "Effective Approaches to Attention-Based Neural Machine Translation," in EMNLP, 2014, pp. 6.

Appendices

Appendix 1:



```python
sim = MatterSim.Simulator()
# Configure the simulator
sim.setCameraResolution(640, 480)
sim.setPreloadingEnabled(True)
sim.setDepthEnabled(False)
sim.setBatchSize(1)
sim.setCacheSize(200)
# Initialize the simulator
sim.initialize()
# Start a new episode
episode = sim.newEpisode(['2t7WUuJeko7'],
['1e6b606b44df4a6086c0f97e826d4d15'], [0], [0])
# Function to move the agent forward for a specified
# number of steps
def move_forward(steps):
    for _ in range(steps):
        # Get the current state
        state = sim.getState()
        # Get the navigable locations (excluding the current viewpoint)
        navigable_locations = state[0].navigableLocations[1:]
        # Choose the first navigable location and move forward
        action = [navigable_locations[1]['ix']]
        heading_change = [0]  # No change in heading (forward)
        elevation_change = [0]  # No change in elevation
        # Make the action
        sim.makeAction(action, heading_change, elevation_change)
# Move the agent forward for 5 steps in different directions
move_forward(5)
```