PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA

OBJECT ATTENTION AND CONTEXTUALIZATION FOR VISION AND LANGUAGE NAVIGATION

BENJAMÍN EARLE

Thesis submitted to the Office of Research and Graduate Studies
in partial fulfillment of the requirements for the degree of
Master of Science in Engineering

Advisor:
ALVARO SOTO

Santiago de Chile, November 2022

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA

OBJECT ATTENTION AND CONTEXTUALIZATION FOR VISION AND LANGUAGE NAVIGATION

BENJAMÍN EARLE

Members of the Committee:

ALVARO SOTO

RODRIGO TORO

JOSÉ MANUEL SAAVEDRA

FELIPE NUÑEZ

Thesis submitted to the Office of Research and Graduate Studies
in partial fulfillment of the requirements for the degree of
Master of Science in Engineering

Santiago de Chile, November 2022

Gratefully to my parents and siblings

ACKNOWLEDGEMENTS

Foremost, I would like to thank my teacher and advisor Álvaro Soto for guiding me through this journey. He taught me everything I know about Machine Learning, believed in my aptitudes, and always pushed me to do better.

I would also like to thank our research group IA Lab, whose members were always open to listen and help whenever needed. Special thanks to Carlos Aspillaga, Felipe del Río, Alain Raymond, Francisca Cattan and Joaquín Ossandon, without whom I would never have gotten this far.

Special thanks to my parents and siblings, who always supported me in my love for science and computers. They have been with me every step of the way, and it's because of them that I had the opportunity to follow a masters degree.

Thanks to my girlfriend and friends, who always knew how to cheer me up when having a tough time in this process.

Finally, thanks to the PUC Computer Science Department, for the great teaching, motivation and friendliness throughout my whole career.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Vision-Language Navigation is a task where an agent must navigate different environments following natural language instructions. This demanding task is usually approached via machine learning methods, training the agent to learn navigation strategies that follow what is said in the instruction and grounding it with what's seen from its environment. However, there is still a gap between human performance and current Vision-Language Navigation models. These instructions usually refer to objects present in the agent's scene, so proper understanding of what's around the agent is necessary to understand where to go and when to stop. This understanding is left to be learned implicitly from the global features of its vision, which are not designed to do object detection. In this work, we propose methods to include and attend to objects during navigation in recurrent and transformer based architectures. We achieve a 1.6% improvement over the base models in unseen environments. But we also see that these models also take advantage of the objects to overfit on seen environments, increasing the gap between the validation seen and unseen splits.

Keywords: Vision-Language Navigation, Deep Learning, Computer Vision, Object Detection, Natural Language Processing, Auxiliary Tasks.

RESUMEN

En la tarea de Navegación por Visión y Lenguaje, un agente debe navegar distintos entornos de acuerdo con una instrucción en lenguaje natural. Esta demandante tarea es comúnmente abordada a través de técnicas de aprendizaje de máquina, los cuales entrenan al agente a aprender estrategias de navegación que siguen lo dicho en la instrucción, aterrizándola a lo que puede ver de su entorno. Actualmente, existe una brecha entre el rendimiento humano y el de modelos de Navegación por Visión y Lenguaje. Estas instrucciones usualmente hacen referencia a objetos que están presentes en el entorno del agente, y el entendimiento de lo que este tiene a su alrededor es necesario para entender hacia donde ir y donde detenerse. Usualmente, este entendimiento se deja para aprender de forma implícita desde las características globales de su visión, las cuales no están diseñadas para detectar objetos. En este trabajo se proponen métodos para incluir y atender objetos durante la navegación del agente con modelos basados en arquitecturas recurrentes o de transformadores. Nuestro método alcanza una mejora relativa de 1.6% sobre los modelos base en entornos desconocidos. A pesar de esto, también se concluye que estos modelos aprovechan la información de objetos para sobreajustar a entornos conocidos, aumentando la brecha entre los conjuntos de validación conocidos y desconocidos.

Palabras Claves: Navegación por Visión y Lenguaje, Aprendizaje Profundo, Visión por Computador, Detección de Objetos, Procesamiento de Lenguaje Natural, Tareas Auxiliares.

1. INTRODUCTION

1.1. Motivation

Recent advances in computer vision and natural language processing using deep learning have had great success in a variety of tasks, such as object recognition and language modeling. Following this, there has been increasing interest in combining both modalities, leading to the creation of new challenges in tasks such as Visual Question Answering (Antol et al., 2015; Singh et al., 2019; Biten et al., 2019), Image Captioning (Mao et al., 2014; X. Chen et al., 2015), Referring Expression Recognition (Kazemzadeh, Ordonez, Matten, & Berg, 2014; Mao et al., 2016), Image Retrieval (Mezaris, Kompatsiaris, & Strintzis, 2003; Wu et al., 2021) and Activity Recognition (Yatskar, Zettlemoyer, & Farhadi, 2016), among others.

With the advent of vision and language related tasks, a natural step is to explore the interaction and communication between an embodied agent and humans. In a way, this leads to the problem of Vision and Language Navigation (VLN), in which a human gives a natural language instruction to the agent, who must then be able to navigate and reach a goal in correspondence to the instruction. A solution to this task leads to better navigation and route planning mechanisms, following human instructions. This would allow for better and safer navigation of an agent, as it would take the route specified by a human who might have prior knowledge of its environment and doesn't want the agent to take a shortest route approach. This has many applications, such as guiding a self-driving car towards your destination or item retrieval within a house or storage area, while taking preferred routes or avoiding potentially dangerous areas.

Recently, a diverse set of open tasks have been proposed for developing and testing agents in VLN. One of the most common tasks is Room-to-Room (R2R) (Anderson et al., 2017), in which the agent must navigate a photo-realistic house environment to reach a goal in accordance to a given natural language instruction. Current state of the art models

reach the goal around 80%[1] of the time, but they take extremely long paths or require an expensive pre-exploration phase, both of which are not realistic in real world applications. Those that have realistic path lengths or don't use pre-exploration only reach their goal around 65%[2] of the time. This shows that the R2R task is an open problem, as humans are able to reach the goal 86% of the time (Anderson et al., 2017) on previously unseen environments and with realistic path lengths.

Current challenges in R2R reside in the alignment between the visual and language inputs, with some research showing that an agent can go blind or not completely take advantage of the visual input and still reach its destination (Hu et al., 2019; W. Zhu et al., 2021; Ossandón et al., 2022). Based on these challenges, this thesis proposes a way to explicitly include objects present in the agent's scene to improve the alignment between objects seen in the environment to those referred to in the given instruction.

1.2. Thesis Overview

Chapter 2 introduces background information required to understand the core concepts of the proposed method. It gives a brief overview of Deep Learning and it's techniques, such as Convolutional Neural Networks, Object Detection, Recurrent Neural Networks, Transformers and Auxiliary Tasks.

Chapter 3 gives an introduction to the Vision and Language Navigation task, and summarizes previous work related to the task. This includes environments, simulators, datasets, models, metrics, and other techniques currently used for this task. It also shows research on current issues with state of the art models.

Chapter 4 gives a detailed description of the proposed method. This includes its intuition, data sources, a description of the base models used for the solution, a description of the proposed module to be included in these models, and their implementation.

[1] According to the leaderboard at https://www.bringmeaspoon.org
[2] Also according to https://www.bringmeaspoon.org

Chapter 5 describes experiments along with the datasets used for them.

Chapter 6 provides quantitative and qualitative results and their analysis, as well as an ablation study of the proposed module.

Finally, chapter 7 presents a conclusion that summarizes the work done, results obtained and their implication for the task. It also describes future work that could be done to extend this work.

2. BACKGROUND INFORMATION

2.1. Deep Learning

Deep Learning methods are representation-learning methods, obtained by composing simple non-linear modules that each transform the representation at one level into a representation at a higher, slightly more abstract level (LeCun, Bengio, & Hinton, 2015). These modules correspond to a layer of perceptrons, followed by an activation function, such as a sigmoid, and its composition is usually called a multi-layer perceptron, or MLP. The key aspect of deep learning is that these representations are not designed by humans, they are learned from data by using a general-purpose learning procedure such as Stochastic Gradient Descent (LeCun et al., 2015).

The core aspects of Deep Learning consist of the design of a network architecture, the collection and processing of data, a loss function and an optimization algorithm (LeCun et al., 2015).

Aside from the MLP architecture, other compositions of perceptron layers have been designed, proving to be successful in a variety of tasks. In the following sections, we'll review some of these which are relevant to this thesis' work.

2.1.1. Convolutional Neural Networks

Convolutional Neural Networks (CNN) are currently one of the most popular Deep Learning architectures in computer vision. These networks are designed to process data that comes in the form of multiple arrays, such as the three color components of a 2D image, each pixel containing the intensity of the red, green and blue colors (LeCun et al., 2015). They are composed of filter banks, which are then passed through the data as a discrete convolution, generating one feature map per filter in the bank (LeCun et al., 2015). A high level diagram of these networks can be seen in Figure 2.1. The advantage

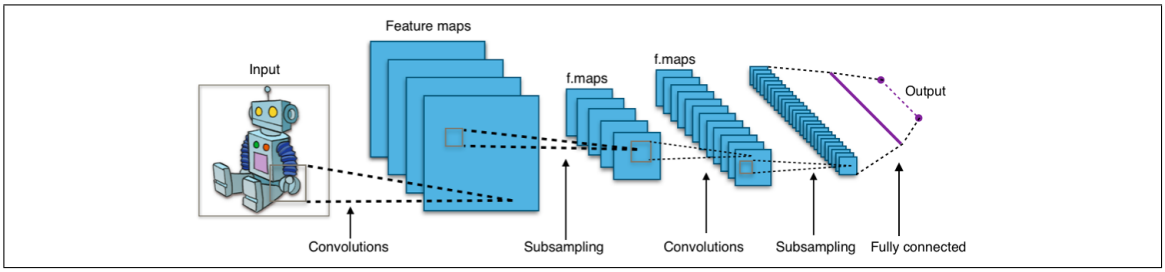


Figure 2.1. Example of CNN architecture. In this example, an image is fed into the network as an input, which produces an output vector that can be used for classification or as a representation for other downstream tasks.

of these networks is that they are, by design, translation invariant, which is desirable when looking for certain patterns present in any location of the given input.

Each of the layers of a CNN detects the most common patterns, which are then used by the following layer to compose them into more complex patterns (Krizhevsky, Sutskever, & Hinton, 2017). For example, take a number recognition task. The first layer would probably detect edges, the second would compose them into segments, the third into parts like lines or circles, and the last layer would recognize the different possible numbers.

2.1.1.1. Object Detection

Object detection is a task that usually involves the use of convolutional neural networks. The goal of this task is to generate bounding boxes within an image in areas where different entities (like persons, chairs or cars) are present and classify them from a set of classes. Object detection architectures use CNNs to regress these bounding boxes and to classify the objects. Examples of these architectures are Faster-RCNN (Ren, He, Girshick, & Sun, 2015) and YOLO (Redmon & Farhadi, 2018). An example of this task can be seen in Figure 2.2.

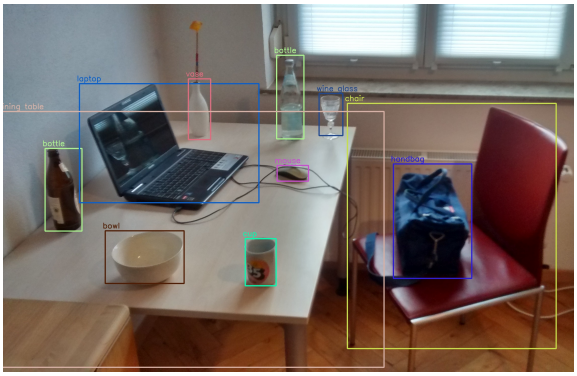


Figure 2.2. Example of the object detection task. Objects in this image have been identified, located and classified. Colored bounding boxes and class labels for each object are shown in the picture.

2.1.2. Recurrent Neural Networks

For tasks that involve sequential inputs and/or outputs, such as text or VLN, Recurrent Neural Networks (RNN) are usually preferred. RNNs process an input sequence one element at a time, maintaining in their hidden units a state vector which contains information of all past elements of the sequence (LeCun et al., 2015). These networks have been found to be very good at predicting the next character in text or the next word in a sentence, but they have also been used in more complex tasks (LeCun et al., 2015).

One of the most popular RNNs are Long Short-Term Memory networks (LSTM), which contain memory cells and gate units, allowing the network to select certain information in their hidden state to remember or forget as it sees fit (Hochreiter & Schmidhuber, 1997). LSTM networks have solved gradient related problems of classic RNNs, which typically tend to vanish or explode over time (Bengio, Simard, & Frasconi, 1994), thus allowing them to properly learn with long sequential data. The architecture of a LSTM cell can be seen in Figure 2.3.

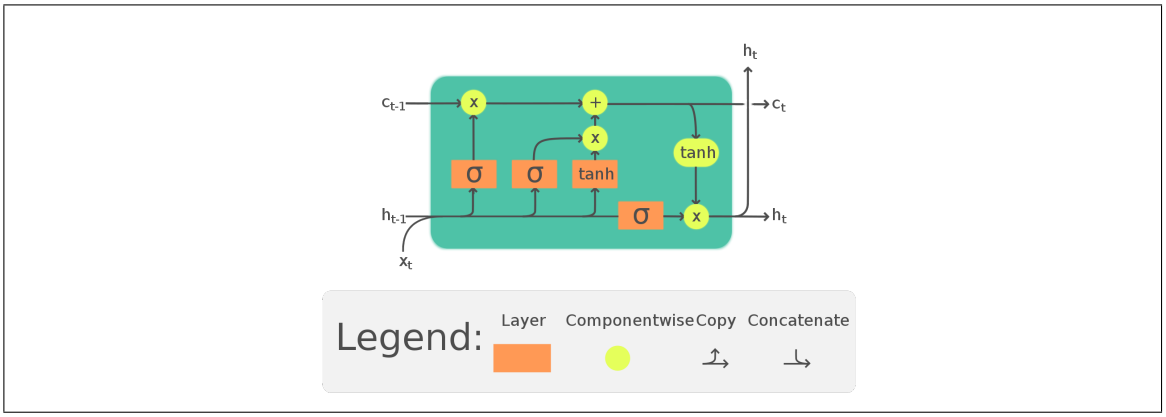


Figure 2.3. LSTM cell architecture. This network takes as previous cell state (h_{t-1}, c_{t-1}) and an input vector x_t, generating a new state (h_t, c_t) to be used in subsequent steps of the RNN. The hidden state h_t is usually also used as a representation for the current time step for other downstream tasks.

2.1.2.1. Transformers

Attention mechanisms have become an integral part of sequence modeling and transduction models in various tasks, usually used in conjunction with RNNs (Vaswani et al., 2017). These mechanisms can be described as a mapping of a query and a set of key-value pairs to an output, with all of these components represented with vectors. The output corresponds to a weighted sum of the values, where the weights are computed via some similarity function between the query and the keys.

Transformer architectures, introduced by Vaswani et al. (2017), eliminates the need for recurrent networks and instead rely entirely on attention mechanisms to draw global dependencies between inputs and outputs. They are significantly more parallelizable than recurrent networks (Vaswani et al., 2017), and have been used on state of the art architectures in various tasks.

A popular transformer based approach for language modeling is BERT (Devlin, Chang, Lee, & Toutanova, 2019), which is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer

to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications.

2.1.3. Auxiliary Tasks

Auxiliary tasks correspond to other tasks parallel to the main objective, and are introduced as extra terms in the loss function, sometimes also including extra parameters within the deep learning model. They have been shown to serve as a regularizer by including an inductive bias to assist in learning useful features for the main task within the shared hidden layers (Y. Liu, Zhuang, Shen, Chen, & Yin, 2019). Once the training is done, these auxiliary tasks are usually removed as they are not required for performing inference on the main objective.

3. RELATED WORK

3.1. Vision-Language Navigation

Recently, a diverse set of tasks have been proposed within the scope of Vision and Language Navigation (VLN), in which an agent must navigate its visual environment given a natural language instruction.

Based on the photo-realistic scans in Matterport3D (Chang et al., 2017), Anderson et al. (2017) introduce the first benchmark dataset for visually-grounded natural language navigation in real buildings, named Room-to-Room (R2R). Subsequently, Qi et al. (2019) introduce REVERIE, also based on Matterport3D, in which the task not only consists on navigation, but also includes identification and localization of a specific object mentioned in the instruction. Following the popularity of the R2R dataset, many biases have been identified in the paths that it labeled, for which Ku, Anderson, Patel, Ie, and Baldridge (2020) introduce a new benchmark dataset, Room-Across-Room (RxR), addressing many of these biases, also including dense spatio-temporal grounding and instructions in Hindi and Telugu, as well as English.

On the other hand, some datasets are based on simulated artificial environments, such as gSCAN (Ruis, Andreas, Baroni, Bouchacourt, & Lake, 2020), focused on evaluating compositional generalization in situated language understanding within a grid-world, and ALFRED (Shridhar et al., 2019), a benchmark for connecting natural language to actions and objects in a simulated environment based on AI2-THOR (Kolve et al., 2017).

3.2. Room-to-Room and Room-Across-Room

The Matterport3D dataset (Chang et al., 2017) includes RGB-D building scale scenes of 90 different home environments, annotated with data regarding present objects, rooms and regions. Based on this dataset, Anderson et al. (2017) developed the Matterport3D Simulator, which simulates its environments and exposes an interface for navigating within

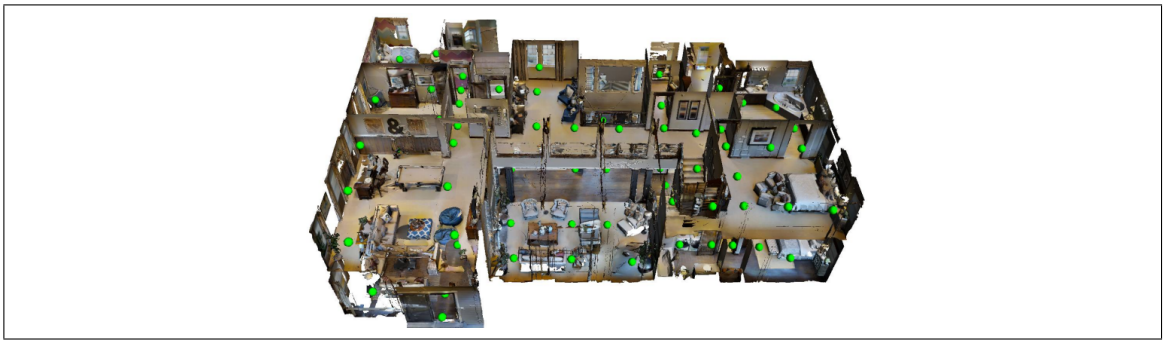


Figure 3.1. Example of a Matterport building (Chang et al., 2017). The building is discretized into viewpoints, which are represented in this image as green nodes. A connectivity graph of these nodes is also included within the Matterport3D metadata.

them. The Room-to-Room dataset (Anderson et al., 2017) uses this simulator and divides the home environments into training and validation (seen and unseen) splits. It also includes 7,189 distinct shortest route paths within these environments, each one with 3 distinct human instructions, totaling 21,567 instructions with an average of 29 words. An extension of the R2R dataset was introduced by Ku et al. (2020), including more and longer paths (no longer the shortest route), longer instructions and spatio-temporal alignment data between instructions and their respective paths.

On these datasets, a set of metrics are proposed to evaluate the performance of agents solving the task:

Path Length (PL) measures the average total distance in meters traversed by the agent until it stops.

Navigation Error (NE) represents the average error in meters between the goal and the point the agent stopped.

Success Rate (SR) measures the percentage of predicted trajectories that end within three meters from the goal.

Success Rate weighted by Path Length (SPL) represents the Success Rate, but the average is weighted by the normalized Path Length, measuring if the agent took an efficient path to the goal.

Oracle Success Rate (OSR) represent the success rate, but taking a successful navigation as one that passes within three meters of the goal at some point in the trajectory.

This work is developed using the Room-to-Room dataset and these metrics for training and evaluating deep learning models.

3.3. Models and Aproximations

Within the framework of the R2R and RxR datasets, a diverse set of architectures, data augmentation strategies and training methodologies are proposed. Originally, Anderson et al. (2017) introduce an LSTM based sequence to sequence neural network agent with an attention mechanism, upon which many future improvements are based.

3.3.1. Data Augmentation

Regarding data augmentation strategies, Fried et al. (2018) introduce a speaker model which synthesizes new instructions and evaluates how well a candidate action sequence explains an instruction. Similarly, Ossandón et al. (2022) present a new method for generating instructions with a focus on objects visible in the agent's environment. Both works show that the agent's generalization and performance increase when trained with these generated instructions.

Tan, Yu, and Bansal (2019) address the problem that agents perform dramatically worse in unseen environments by introducing an environment dropout data augmentation strategy, which improves upon environment variability.

C. Liu, Zhu, Chang, Liang, and Shen (2021) introduce a Random Environmental Mixup method, which generates cross connected house scenes as augmented data via mixing environments.

Finally, new datasets have been introduced, which contain longer paths that are not the shortest route. This is achieved by joining Room-to-Room routes, generating the Room-4-Room dataset (Jain et al., 2019), as well as the Room-6-Room and Room-8-Room datasets (W. Zhu et al., 2020).

3.3.2. Auxiliary Tasks

Many auxiliary tasks are also proposed to improve grounding and training performance. Ma, Lu, et al. (2019) introduce a progress monitoring task, to ensure the grounded instruction correctly reflects the navigation progress. F. Zhu, Zhu, Chang, and Liang (2019) use a similar task to the progress monitor, an angle prediction task to learn orientation from the visual features, a trajectory retelling task, making the agent reconstruct the instruction from the trajectory in an end to end fashion, and a cross modal matching task, making the agent identify whether a given instruction is consistent with the trajectory, similarly to LXMERT (Tan & Bansal, 2019). Manterola (2021) implements a scene recognition task, classifying the agent's current room for every timestep, in order to build a more robust representation of the world. Finally, Qi et al. (2021) use the progress estimator, a direction prediction task, and a scene recognition task, which aims to classify the corresponding room for the goal and next timestep.

3.3.3. Navigation Strategies

Some approaches address different navigation strategies to handle uncertainty and robustness upon erroneous actions. Ma, Wu, AlRegib, Xiong, and Kira (2019) take advantage of the progress monitor auxiliary task as a learnable heuristic, allowing the agent to rollback to a previous state if the progress estimation decreases by more than a predefined threshold. Wang, Wang, Liang, Xiong, and Shen (2021) maintains a memory of all seen

nodes during its navigation and allows action selection based on a global action space instead of a local one (i.e. select a node from all candidates previously seen), making a robust planning over the frontier of exploration. Finally, Wang, Wang, Shu, Liang, and Shen (2020) introduce a method for choosing to explore a node, gather information and update its representation to resolve ambiguity and uncertainty.

3.3.4. Pre-Training

Other approaches take advantage of pre-training strategies to improve a model's representation of the world and instructions in order to improve navigation performance when training with the full dataset. Majumdar et al. (2020) pre-trains a transformer based architecture with image-text pairs from the web, allowing the agent to learn better correlation between its visual and linguistic inputs. Shen et al. (2021) use contrastive learning to pre-train a visual encoder that takes into account a natural language input, and uses it in a series of vision and language tasks, such as R2R. S. Chen et al. (2021) implement a language-model like pre-training for its BERT based architecture, including tasks like Masked Language Modeling, Masked Region Modeling, Single Step Action Prediction/Regression and Instruction Trajectory Matching. Finally, W. Zhu et al. (2020) use curriculum-based reinforcement learning to maximize rewards on navigation tasks with increasingly longer instructions.

3.3.5. Transformer based models

While most models are RNN based, other approaches use transformers as their encoder-decoder architecture. Majumdar et al. (2020) and Qi et al. (2021) implement a recurrent modification of a BERT based architecture. On the other hand S. Chen et al. (2021) also use a BERT based model, but propose a method of keeping the navigation context based on all previously seen panoramas instead of a recurrent modification of the architecture.

3.3.6. Extra inputs to models

Aside from instructions and panoramic views, REVERIE includes object bounding boxes used in its particular task. Hong, Wu, Qi, Opazo, and Gould (2020) use a transformer based architecture on R2R and REVERIE, only including object data when solving the REVERIE task. On the other hand, Qi et al. (2021) replace the panoramic views in R2R with the objects present in the scene, as annotated in REVERIE, but when solving the R2R task. On the other hand, F. Zhu, Zhu, Long, Chang, and Liang (2020) include objects to AuxRN (F. Zhu et al., 2019), treating them similarly to the panoramic images already used by the architecture. Also related to objects, Hu et al. (2019) include the labels, colors and bounding box coordinates, without the object's features, extracted by an object detector as inputs to the model, using a mixture of experts approach to further enrich the model's inputs. These models include objects as an additional input to the network, but without any special processing or contextualization of them. No work was found which contextualized objects to enrich their representation and use in action prediction.

Finally, Gao et al. (2021) include a knowledge base from ConceptNet (Speer, Chin, & Havasi, 2016) to include this external knowledge into a scene memory module, used for action prediction.

3.3.7. Current Limitations

Over the years, a diverse set of limitations have been detected among state of the art models tackling R2R and similar tasks. Hu et al. (2019) find that, when removing vision from an agent (i.e. setting its visual input to zero), the performance hardly drops, and sometimes it improves on the unseen validation set, showing that they don't use visual inputs in any generalizable way.

Similarly, W. Zhu et al. (2021) investigate the agent's perception of multimodal inputs. They find that removing object references in the instruction greatly reduces the agent's

performance. On the other hand, when masking out objects in the agent's vision, its performance drop isn't as significant. They conclude that agents are not able to properly align objects in their visual perception to those present in the instruction.

4. PROPOSED METHOD

4.1. Object contextualization

Current state of the art models have a way of crossing the information coming from the multi-modal inputs, usually done by some form of attention mechanism. This, being done with the global features extracted via a ResNet-152 convolutional network (He, Zhang, Ren, & Sun, 2015), gives current state of the art results. But, according to the findings of Hu et al. (2019) and W. Zhu et al. (2021), these models are not really taking advantage of the information present in their visual perception. Also, these visual features are designed for image classification and not object detection, so their encoded information may be limited by this.

This work focuses on including object specific information as an input to these models in a way that can complement the global visual information, improving the alignment between objects mentioned in the instruction and objects seen in the environment.

4.2. Intuition

Instructions in the R2R dataset rely on references to objects present in the scenes the agent traverses. As such, it is exceedingly important for the agent to have a proper knowledge and understanding of the objects it sees and how they align with the given instruction. If an agent is given an instruction that ends with “Go straight until you reach the table and wait there”, knowing where the table is in relation to the agent is a rich source of information, instantly letting it know where to go or if it has reached its destination. Properly being able to align the objects mentioned in the instruction with those seen in the agent’s environment should allow it to improve its navigation performance, as it can use these references to monitor its progress and reduce uncertainty while navigating.

4.3. Data Sources and Preprocessing

This work uses the data available for the R2R (Anderson et al., 2017) task, which provides instructions for paths within home environments made available in Matterport3D (Chang et al., 2017). This includes 7,189 shortest route paths within 90 different home environments. Each path contains 3 human instructions, totaling 21,567 instructions with an average of 29 words.

We also use the data generated by Ossandón et al. (2022), which includes 178,300 sampled paths, each with its own instructions. These have a special focus on objects present within the route as reference for navigation actions. The generated data is available only for training and validation unseen splits.

In order to make the training and evaluation of models more efficient, Anderson et al. (2017) provide pre-computed features for the visual scenes present in Matterport3D. These features are extracted from the mean pooling layer after the last convolution of a ResNet-152 (He et al., 2015) network, pre-trained on ImageNet (Russakovsky et al., 2015), resulting in 36 2048-dimensional vectors for each scene. On the other hand, S. Chen et al. (2021) extract pre-computed features from the last layer of a visual transformer, resulting in a 768-dimensional feature vector for each image in the panoramic.

Matterport3D also has annotations on the objects present in the scene, including their location and label. For the proposed modification of the EnvDrop model, the last convolutional map of the same ResNet-152 was extracted in order to obtain features for each of these objects. Based on the object's location, it's position in the corresponding image was computed, which was then used to find its proportional position within the convolutional map. A region of interest (ROI) pooling method is used to extract a $2 \times 2 \times 2048$ tensor, centered on the position of the object in the convolutional map, as the features of a specific object.

For the proposed modification of the HAMT model, object bounding boxes were computed based on the orientation and vertical field of view of each object. The corresponding image to each bounding box was then used to compute the object's features by passing it through the same visual transformer used to precompute panoramic features. Object and room labels are also kept for an auxiliary classification task.

4.4. Problem Setup

The R2R task (Anderson et al., 2017) requires an agent to find a route from a start viewpoint to a target viewpoint according to a given instruction I. At each timestep t, the agent's observation consists of a panoramic view O_t discretized into 36 single views[1] $V_t = \{v_{t,i}\}_{i=1}^{36}$. Each of these views $v_{t,i}$ is an RGB image, accompanied by its heading $\theta_{t,i}$ and elevation $\varphi_{t,i}$ in radians. The agent is also given a set K of navigable viewpoints $C_t = \{v_{t,k}\}_{k \in K}$, which correspond to $|K| - 1$ reachable and visible locations from the current viewpoint, as well as the stop action. This problem can also be thought as a Partially Observable Markov Decision Process, where the states consist of the panoramic views O_t, the actions are the navigable viewpoints C_t and the model's goal is to choose actions at each timestep that maximizes its expected future discounted reward by learning to predict the conditional transition probabilities $p_t(a_{t,k})$ between states.

As is usually done, views are preprocessed through a ResNet-152 (He et al., 2015), resulting in features $f_{t,i} = ResNet(v_{t,i})$. For HAMT (S. Chen et al., 2021), views are preprocessed through a visual transformer, resulting in features $f_{t,i} = ViT(v_{t,i})$.

In this work, we also add objects visible from the current viewpoint $O_t = \{o_{t,i}\}_{i=0}^{|O_t|}$, which correspond to the pixels within bounding boxes of these objects on the 36 views of the panorama.

[1]The panoramic view covers $360°$ horizontally and $90°$ vertically. Each single view covers $30°$ horizontally and vertically, resulting in 3 elevation and 12 headings.

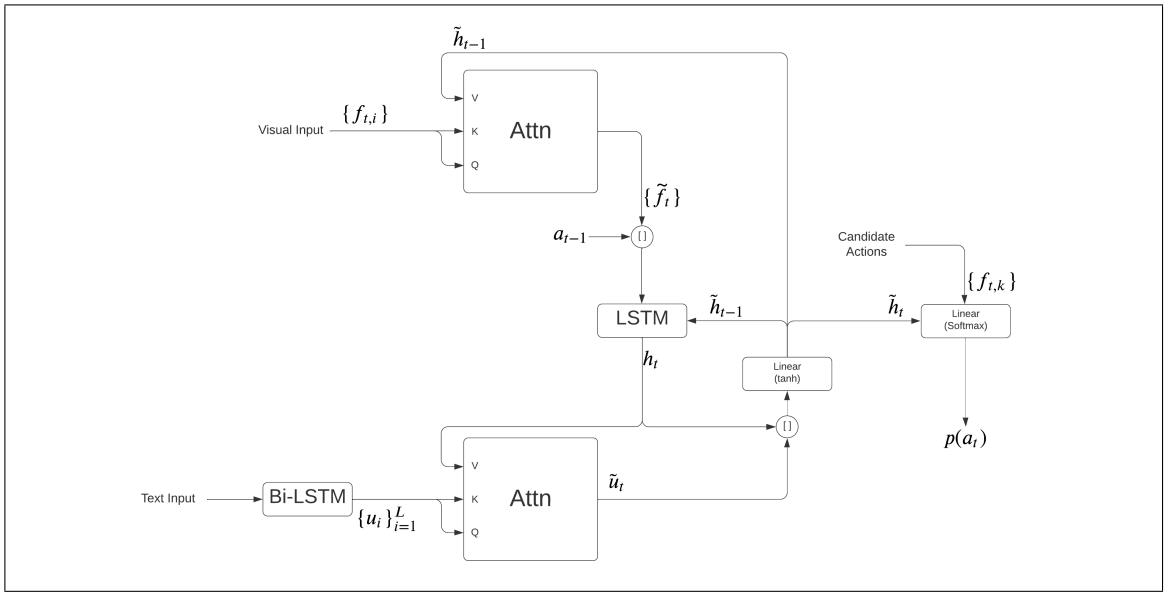


Figure 4.1. EnvDrop model architecture. This model consists of an encoder-decoder network. The encoder takes a text instruction and produces the textual context u_i. The decoder takes the visual input and candidates in each timestep and uses the textual context to decode the current timestep's probability for each candidate action $p(a_t)$.

4.5. EnvDrop Model

Our work builds over the EnvDrop model (Tan et al., 2019). A diagram of this model can be seen in Figure 4.1. They implement an encoder-decoder model, where the encoder is a bidirectional LSTM with an embedding layer for each word token w_i in the instruction I:

$$\hat{w_i} = \text{embedding}(w_j) \tag{4.1}$$

$$u_1 \ldots, u_L = \text{Bi-LSTM}(\hat{w}_1, \ldots, \hat{w}_L) \tag{4.2}$$

The decoder of the agent corresponds to an attentive LSTM. At each decoding step t, the view features $\{f_{t,i}\}$ are attended, resulting in the attentive visual feature $\tilde{f}_t$:

$$a_{t,i} = \text{softmax}_i(f_{t,i}^\top \, W_F \, \tilde{h}_{t-1}) \tag{4.3}$$

$$\tilde{f}_t = \sum_i \alpha_{t,i} f_{t,i} \tag{4.4}$$

The input of the decoder corresponds to the concatenation of the attentive visual feature $\tilde{f}_t$ and an embedding of the previous selected action $\tilde{a}_{t-1}$. The hidden output h_t of the decoder LSTM is combined with the attentive instruction feature $\tilde{u}_t$ to form an instruction-aware hidden output $\tilde{h}_t$:

$$h_t = \text{LSTM}\Big(\big[\tilde{f}_t; \tilde{a}_{t-1}\big], \tilde{h}_{t-1} \Big) \tag{4.5}$$

$$b_{t,j} = \text{softmax}_j(u_j^\top \, W_U \, h_t) \tag{4.6}$$

$$\tilde{u}_t = \sum_j \beta_{t,j} u_j \tag{4.7}$$

$$\tilde{h}_t = \tanh\big(W \big[\tilde{u}_t; h_t\big]\big) \tag{4.8}$$

Finally, the probability $p_t(a_{t,k})$ of moving to the k-th navigable viewpoint is calculated as the softmax of the alignment of the navigable feature $f_{t,k}$ and the instruction-aware hidden input:

$$p_t(a_{t,k}) = \text{softmax}_k(f_{t,k}^\top \, W_G \, \tilde{h}_t) \tag{4.9}$$

4.5.1. Object and Language Contextualization Module (OLC)

In this work, an object and language contextualization (OLC) module is designed to be easily integrated into existing recurrent based models. It has as an input the object

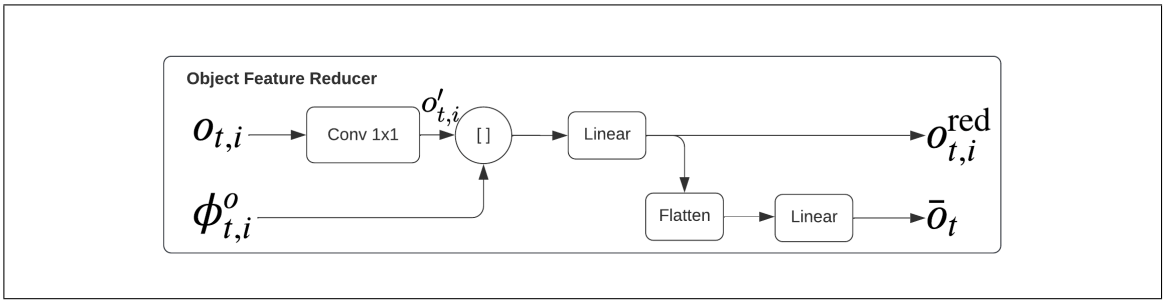


Figure 4.2. Object Feature Reducer. It takes as an input the current viewpoint's objects' feature vector o_t and orientation ϕ^o_t. It produces two outputs: a summary vector $\bar{o}_t$ which represents the objects present in the current viewpoint, and a size-reduced feature vector $o^{\text{red}}_{t,i}$ for each object.

features $O_t = \{o_{t,i}\}$ and orientations $\phi^o_{t,i} = (\theta^o_{t,i}, \varphi^o_{t,i})$, a navigation context c_t for the current timestep[2], and the navigable viewpoints orientations $\phi^f_{t,j} = (\theta^f_{t,j}, \varphi^f_{t,j})$.

4.5.2. Object Feature Reducer

Object features come from ROI-aligned features of the last convolutional map used to compute $f_{t,i}$. In order to reduce their size and flatten the features, they are passed through a 1x1 convolution, combined with their orientation and processed by a linear layer, as shown in Figure 4.2. This results in the reduced object features $o^{\text{red}}_{t,i}$:

$$o'_{t,i} = \text{Conv}_{1\times1}(o_{t,i}) \tag{4.10}$$

$$o^{\text{red}}_{t,i} = W_r\ [o'_{t,i}; \phi^o_{t,i}] \tag{4.11}$$

Where W_r is a learnable parameter.

In order to be able to use object information to include in the decoder's input, an object summary $\bar{o}_t$ is also computed via flattening the collection of object features and passing through a linear layer:

[2]In the case of the EnvDrop model, this corresponds to the instruction-aware hidden output $\tilde{h}_t$.

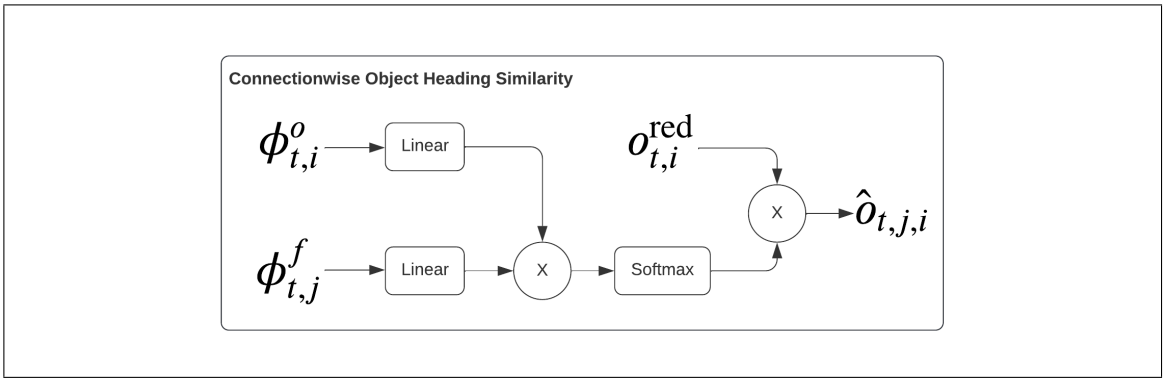


Figure 4.3. Connectionwise Object Heading Similarity. Computes the similarity between the orientation of an object $\phi^o_{t,i}$ and a navigable viewpoint $\phi^f_{t,i}$, outputting $\hat{o}_{t,j,i}$: a weighted sum of the object features $o^{\text{red}}_{t,i}$ for each navigable viewpoint.

$$\bar{o}_t = W_s \text{ flatten}(o^{\text{red}}_{t,i}) \tag{4.12}$$

Where W_s is a learnable parameter. This summary contains information of all the objects present in the agent's scene.

For example, this summary can be used in the EnvDrop model by including the object summary in the computation of the decoder's hidden state h_t in Equation 4.5:

$$h_t = \text{LSTM}\Big(\big[\tilde{f}_t; \tilde{a}_{t-1}; \bar{o}_t\big], \tilde{h}_{t-1}\Big) \tag{4.13}$$

4.5.3. Connectionwise Object Heading Similarity

In order to explicitly give the agent the capability to attend to the objects in an independent manner for each navigable viewpoint, they are attended according to their orientations. The module, shown in Figure 4.3, aims to do this. In this way, for each navigable viewpoint j, a score $\gamma_{t,j,i}$ is computed for each object i. Then, a representation for each object in each viewpoint ($\hat{o}_{t,j,i}$) is computed by weighing the objects with this score:

$$\gamma_{t,j,i} = \text{softmax}_i((\phi^f_{t,j})^\top \, W_\gamma \, \phi^o_{t,i}) \tag{4.14}$$

$$\hat{o}_{t,j,i} = \gamma_{t,j,i} \, o^{\text{red}}_{t,i} \tag{4.15}$$

Where W_γ is a learnable parameter. The vector $\hat{o}_{t,j,i}$ contains information of the i-th object's feature, orientation and relative orientation to the j-th navigable viewpoint.

4.5.4. Connectionwise Object Attention

Finally, this module is used to obtain a representation of the relevant objects for each navigable viewpoint. An attention mechanism is used to contextualize objects for a navigable viewpoint according to the navigation context input c_t, resulting in a representation for each navigable viewpoint $\tilde{o}_{t,j}$:

$$\phi_{t,j,i} = \text{softmax}_i(\hat{o}_{t,j,i} \, W_\phi \, c_t) \tag{4.16}$$

$$\tilde{o}_{t,j} = \sum_i \phi_{t,j,i} o_{t,j,i} \tag{4.17}$$

This representation can be used to assist in the selection of the next navigable action. For example, in the EnvDrop model, the probability $p_t(a_{t,k})$ of moving to the k-th navigable viewpoint (shown in equation 4.9) is now computed as:

$$p_t(a_{t,k}) = \text{softmax}_k\big([f_{t,k}; \tilde{o}_{t,k}]^\top \, W_G \, \tilde{h}_t\big) \tag{4.18}$$

A complete diagram of the proposed module is shown in Figure 4.4, and it's integration with the base model is shown in Figure 4.5.

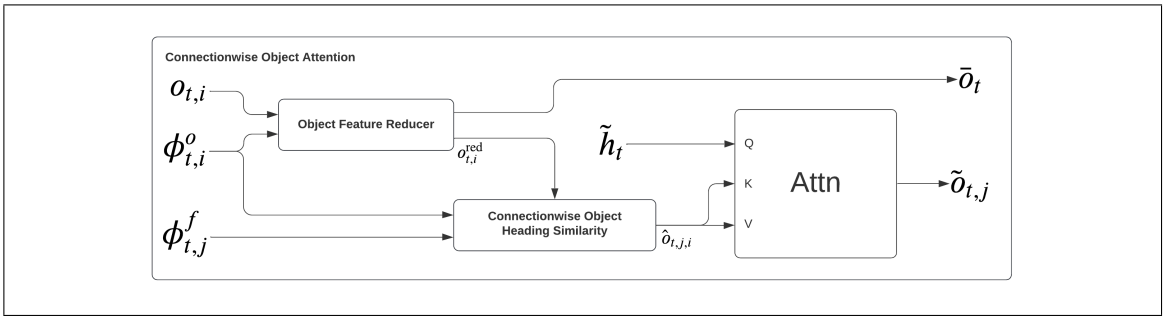


Figure 4.4. A complete diagram of our Object and Language Contextualization module. It takes as input the object features and orientations, a navigation context input and navigable viewpoint orientations. Its output is a summary representation of the objects seen in the current panorama, and a representation of relevant objects for each navigable viewpoint, contextualized with the navigation context.

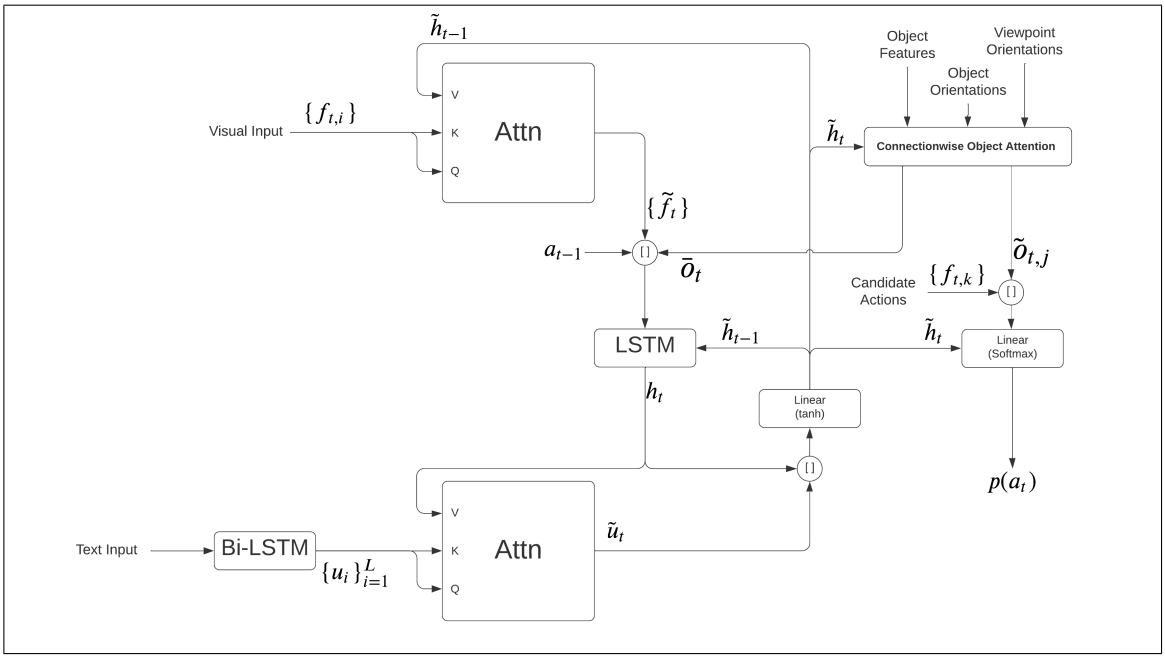


Figure 4.5. A diagram of our module integrated within the EnvDrop model. It uses the instruction-aware hidden state as navigation context, and its outputs are integrated into the decoder's input ($\bar{o}_t$) and action selection ($\tilde{o}_{t,j}$).

4.5.5. Object Classification Auxiliary Task

If object labels $o^*_{t,i}$ are present as an input. An optional auxiliary task can be included within the module. This task aims to classify these objects into their correct category, further grounding the differences between object classes. This is done via two sequential linear layers over the reduced object features $o^{\text{red}}_{t,i}$, followed by a softmax activation. This results in the probability $p_o(o_{t,i})$ for each object to be of a certain class. The resulting loss L_{obj} is then added to the base model's loss with a weight λ_{obj}.

$$L_{\text{obj}}(o_{t,i}, o^*_{t,i}) = -\sum_j o^*_{t,i,j} \ln\left(p_o(o_{t,i,j})\right) \tag{4.19}$$

4.6. HAMT Model

This work also builds over the HAMT model (S. Chen et al., 2021). A diagram of this model can be seen in Figure 4.6. They implement a BERT-like transformer architecture, which takes the instruction, panoramic features, and history of panoramic features as input, each to its own transformer. Then, the output of these transformers are passed through a cross-modal transformer and an action prediction head to predict the agent's next action.

They include a different token embedding for inputs of type text, history, and observation. The observation transformer also includes a navigation embedding indicating if a view is navigable, non-navigable or the stop action. The history transformer also includes a step embedding, which is similar to a positional embedding in that it indicates the timestep of each history panoramic input.

For the text input, the model receives a word embedding w, positional embedding E^P_i and a type embedding E^T_0.

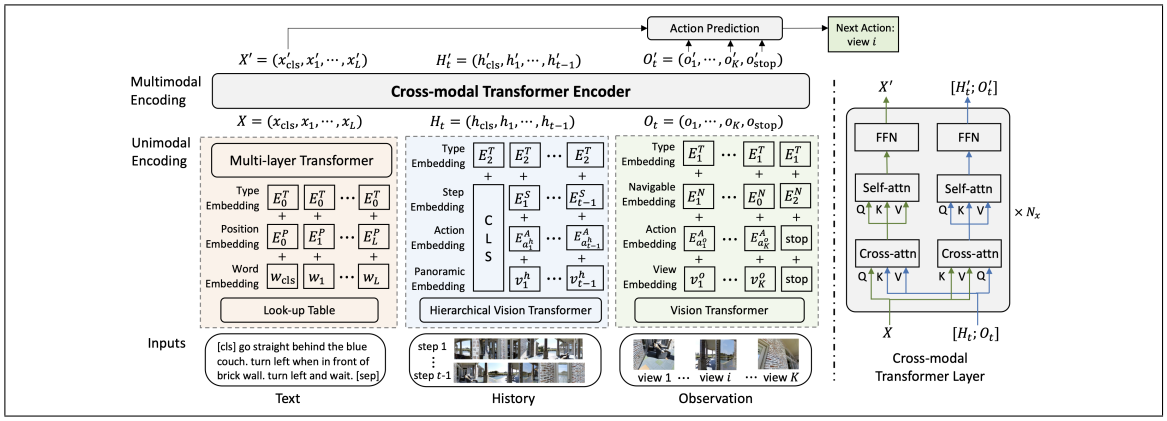


Figure 4.6. HAMT model architecture (S. Chen et al., 2021). It uses three transformers to process the three inputs taken by the network. The output of these transformers are then combined and contextualized via a cross-modal transformer. This generates a cross-modal vector used to predict the agent's next action.

For the observation input, the model receives features of the panoramic features, pre-computed using a Visual Transformer. It uses the visual features v_i, relative orientation $E^A_{a_i^0}$, navigable embedding $E^N_{o_i}$ and token type embedding E^T_1

For the history input, each timestep panoramic is represented by a feature v_i^h computed via their Hierarchical History Transformer. It uses this history feature v_i^h, action orientation embedding $E^A_{a_i^h}$, step embedding E^S_i and token type embedding E^T_2.

Finally, the result of passing these inputs through their own transformers is passed through a cross-modal transformer resulting in output embeddings $X' = (x'_{cls}, x'_1, ..., x'_L$, $H'_t = (h'_{cls}, h'_1, ..., h'_{t-1})$ and $O'_t = (o'_1, ..., o'_K, o'_{STOP})$ for tokens in text, history and observation, respectively.

These tokens are then used for a variety of pre-training tasks, such as Masked Language Modeling, Masked Region Modeling, Instruction Trajectory Matching, Single Step Action Prediction, Single Step Action Regression and Spatial Relationship Prediction.

Finally, after pre-training, the model is fine-tuned reusing the Single Step Action Prediction head with imitation and reinforcement learning approaches.

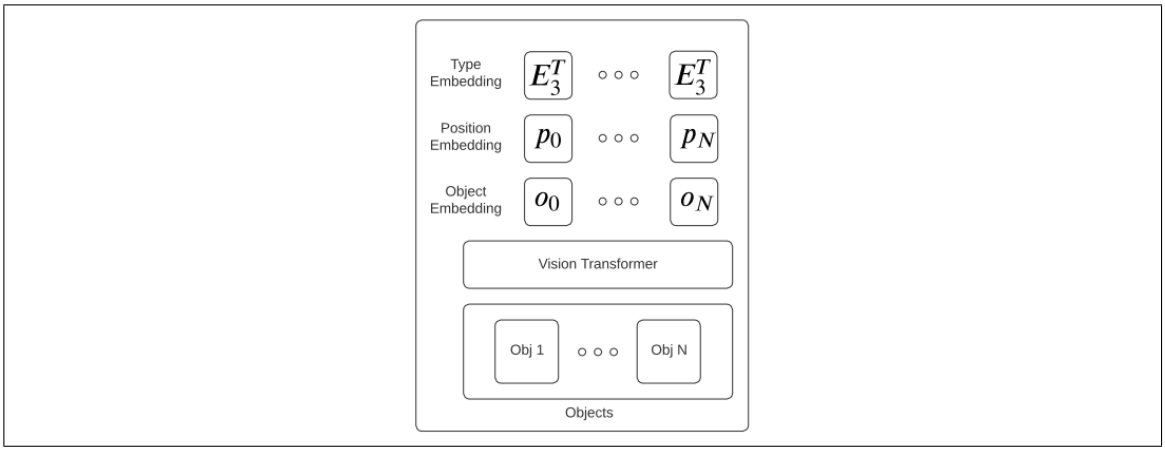


Figure 4.7. Object transformer. Similar to the visual transformer, it takes as an input the objects' visual features, orientations and their corresponding position and type embedding. Its output is a representation for each object, which will be included in the cross-modal transformer to further contextualize the other inputs.

4.6.1. Object Transformer

In order to include object features in HAMT, a new transformer is added to the architecture. This transformer, as seen in Figure 4.7, takes as an input the features o_i of the objects present in the current scene. Each object includes a position embedding p_i, which indicates the relative orientation of the object to the agent, and an object type embedding E_3^T.

The results of this object transformer will also be included in the cross-modal transformer, in conjunction with the text, history and observation inputs, producing output embeddings $Q'_t = (q'_1, ..., q'_N)$. In this way, the information encoded in the other embeddings will also include information of the objects present in the scene. Figure 4.8 shows HAMT with the proposed module.

4.6.2. Object Auxiliary Tasks

In order to adopt the pre-training strategy used in HAMT, two auxiliary tasks are implemented to be performed with object output embeddings Q'_t.

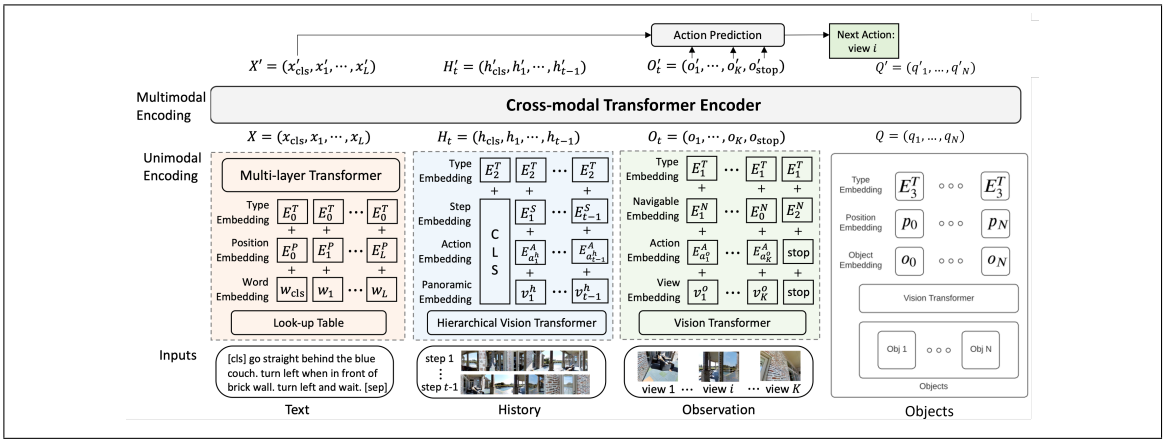


Figure 4.8. A diagram of our module included in HAMT. It extends the transformer architecture with an object transformer, which helps contextualize the instruction, history, and observations.

Similar to Section 4.5.5, an object classification task is implemented, in which the model must be able to predict the object's label as annotated in the Matterport dataset. This is done via two sequential linear layers over the attended object features $q'_{t,i}$, followed by a softmax activation. This results in the probability $p_o(q'_{t,i})$ for each object to be of a certain class. The resulting loss L_{obj} is then added to the base model's loss during pre-training.

$$L_{\text{obj}}(o_{t,i}, o^*_{t,i}) = -\sum_j o^*_{t,i,j} \ln\left(p_o(o_{t,i,j})\right) \tag{4.20}$$

Using a summary of the objects present in the scene, a room classification task is implemented, in which the model must be able to predict the room described by the objects shown to it. This is done via computing the average of the object features and passing it through two sequential linear layers followed by a softmax activation. This results in probabilities $p_r(\{o_{t,1}, ..., o_{t,N}\})$. The computed loss L_{room} is then added to the base model's loss during pre-training.

$$L_{\text{room}}(\{o_{t,1}, ..., o_{t,N}\}, r^*_t) = -\sum_i r_{t,i} \ln\left(p_r(\{o_{t,1,i}, ..., o_{t,N,i}\})\right) \tag{4.21}$$

4.7. Relation between both models

HAMT's approach is based on the use of transformers and pre-training tasks, where our extension builds upon this by adding objects to the transformer architecture and pre-training tasks. On the other hand, EnvDrop's approach is similar to HAMT, but less generalized. Our extension of EnvDrop is designed to be included in a non-transformer architecture, while maintaining the benefits of attention and an auxiliary task (no longer pre-training) over the object inputs. In this way, our OLC module can be thought of as similar to the approach taken for HAMT.

5. EXPERIMENTS

5.1. Dataset

For the experiments, we use the R2R (Anderson et al., 2017) dataset and the augmented data generated by Ossandón et al. (2022).

Due to the high resource requirements for training on the R2R dataset with all present objects, we do experiments with a reduced set of environments and objects[1]. We use 30% of environments from the training and validation seen splits, and all the environments in the validation unseen split. Regarding objects, we uniformly sample a maximum of 32 objects for each viewpoint, as this was the maximum amount we were able to use with the available GPUs[2].

Because of the efficiency of transformer architectures, the full dataset is used for experiments done on the HAMT model.

5.2. Experimental design

As shown in section 4, two different existing models are modified to include an object contextualization mechanism. The same training procedure and hyperparameters as in their original papers will be used, as detailed in this section.

5.2.1. EnvDrop experiments

We implement the described module on the EnvDrop model (Tan et al., 2019). Aside from the baseline, two different variants are used, one which includes object summary $\bar{o}_t$ in the recurrent decoder and the other where it isn't included. These models are trained with the same hyperparameters as those in the original paper.

[1] Selected environments and objects are listed in appendix A

[2] All experiments were done with either an Nvidia 1080Ti (11.2 GB) or TitanRTX (24.2 GB) GPU

The models are optimized with the RMSProp algorithm and a learning rate of 10^{-4} for 150000 epochs with a batch size of 64. When the proposed module is used, 32 objects are sampled from those present in the scene, the module outputs features with a size of 256, and the auxiliary task weight is 0.1.

5.2.2. HAMT experiments

We also implement the HAMT model (S. Chen et al., 2021) with the object transformer and auxiliary tasks. This model is trained in two stages: pre-training and fine-tuning. First, we pre-train two versions of HAMT using R2R instructions, one with objects and the other without. Then, in the fine-tuning stage, four models are fine-tuned. For each pre-trained model, two models are generated, one fine-tuned with R2R instructions, and the other with the craft instructions generated by Ossandón et al. (2022). These models are trained with the same parameters shown in the original HAMT paper (S. Chen et al., 2021).

Pre-training is optimized with the AdamW algorithm, using betas 0.9 and 0.98, and a learning rate of $5 \cdot 10^{-5}$ with a warm-up of 10000 episodes. The pre-training lasts for 200000 episodes with a batch size of 16, using the same task mix ratio as the original paper. When including objects, the object transformer has the same hyperparameters as the visual transformer, while room and object tasks are included with a weight of 1 in the mix ratio.

For fine-tuning, we use the same training parameters as in the original HAMT paper, and the architecture matches that of the pre-training phase. The training lasts for 300000 epochs with a batch size of 8. The model is optimized with the AdamW algorithm and a learning rate of 10^{-5}.

5.3. Evaluation Metrics

Results are measured using the described metrics in Section 3.2 on the validation seen and unseen splits. The main metric to compare results is success rate (SR) on the validation

unseen splits. Path length (PL), navigation error (NE), success rate weighted by path length (SPL) and oracle success rate (OSR) are also included in the results for comparison.

6. ANALYSIS

6.1. Cuantitative Results

6.1.1. EnvDrop

Table 6.1 shows results for EnvDrop models trained with R2R original instructions. By including object information during training and evaluation of the EnvDrop model, we achieve a 6.9% absolute improvement in validation seen, and a 1.6% absolute improvement in validation unseen. This shows that objects in the scene can provide useful information to the agent during navigation.

Results also show an increase of 5.7% in the generalization breach between validation seen and unseen when including objects. This may be because the model is overfitting towards objects present in seen environments, and has more difficulty taking advantage of new objects in unseen environments.

Taking into account the a reduced set of training environments used, we can see that including object information in the decision process of the agent helps in data efficiency, obtaining better results than the baseline in this reduced data situation.

Table 6.1. Results for trained EnvDrop models using R2R instructions. * are reproduced results.

Model	Validation Seen					Validation Unseen				
	PL (↓)	NE (↓)	SR (↑)	SPL (↑)	OSR (↑)	PL (↓)	NE (↓)	SR (↑)	SPL (↑)	OSR (↑)
EnvDrop*	10.99	6.05	45.5	43.8	54.8	9.28	6.68	39.4	36.3	**46.9**
EnvDrop + OLC (w/o summary)	**10.84**	**4.98**	**52.8**	**50.7**	**60.3**	**8.89**	**6.51**	**41.0**	**37.4**	46.2

On the other hand, when training with craft instructions generated by Ossandón et al. (2022), Table 6.2 shows an increased improvement gain than when training with R2R instructions, totaling a 2.3% absolute improvement over the base model. This is because craft instructions have a greater focus on orienting the agent via objects than the instructions from the original R2R dataset. This indicates that, when there's a strong alignment

between objects mentioned in the instruction and those present in the scene, the inclusion of object information can give important cues to the agent during its navigation.

Table 6.2. Results for trained EnvDrop models using Craft instructions. These instructions don't have data for the validation seen environment. * are reproduced results.

Model	Validation Unseen				
	PL (↓)	NE (↓)	SR (↑)	SPL (↑)	OSR (↑)
EnvDrop*	9.35	5.2	49.7	47.6	54.5
EnvDrop + OLC (w/o summary)	**9.31**	**4.96**	**52.0**	**49.8**	**57.1**

6.1.2. HAMT

Table 6.3 shows results for HAMT models trained with R2R original instructions. By including object information into the cross modal transformer, we see an absolute improvement of 7.34% in validation seen and of 0.94% in validation unseen. This shows that objects can give important cues to the agent during navigation. Regardless, this improvement is particularly small in validation unseen. This may be due to the other pre-training tasks the agent is pre-trained with, many of which require the agent to have a better visual representation of its environment. Because of this, this extra visual information may not be as useful as expected.

Results also show a big increase in the generalization gap. Our model obtains an increase of 6.4% in the difference between validation seen and unseen, with respect to the baseline. Once again, this shows a possible increase in overfitting to seen environments through the objects available within them.

Table 6.3. Results for trained HAMT models using R2R instructions. * are reproduced results.

Model	Validation Seen					Validation Unseen				
	PL (↓)	NE (↓)	SR (↑)	SPL (↑)	OSR (↑)	PL (↓)	NE (↓)	SR (↑)	SPL (↑)	OSR (↑)
HAMT*	12.82	3.99	61.02	56.93	68.95	**12.16**	**4.18**	58.02	53.03	65.56
HAMT + Object Transformer	**11.34**	**3.59**	**68.36**	**64.47**	**75.32**	12.37	4.45	**58.96**	**53.36**	**66.75**

On the other hand, Table 6.4 shows results on craft instructions in validation unseen. In this object focused instructions, we achieve a 0.89% absolute improvement over the baseline. Contrary to the results seen with the EnvDrop model, the improvement is slightly lower than with R2R instructions. This reinforces the hypothesis that the other pre-training tasks may already take the role of making the agent more object aware and their alignment with object focused instructions.

Table 6.4. Results for trained HAMT models using Craft instructions. These instructions don't have data for the validation seen environment. * are reproduced results.

Model	Validation Unseen				
	PL (↓)	NE (↓)	SR (↑)	SPL (↑)	OSR (↑)
HAMT*	**11.45**	**3.48**	63.73	59.15	71.39
HAMT + Object Transformer	11.88	3.39	**64.62**	**60.20**	**72.67**

6.2. Ablation Study

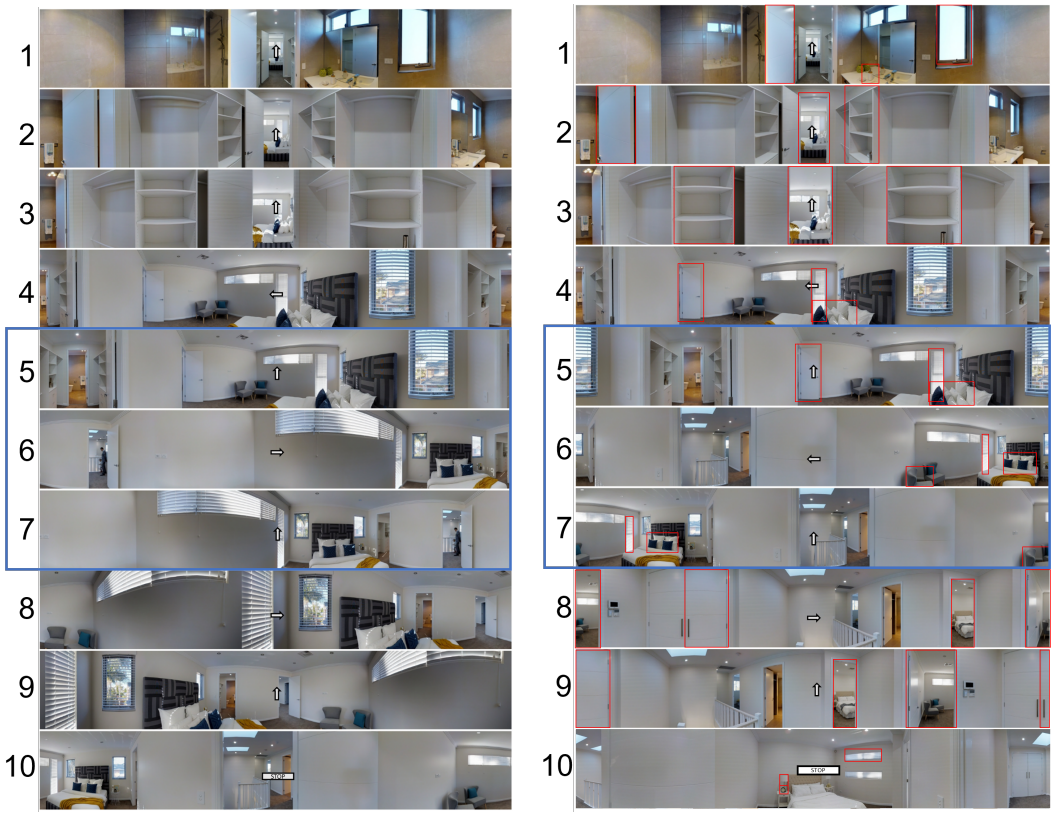
Table 6.5 shows results when also including the object summary described in Equation 4.12. Results are slightly lower when including this summary in the recurrent step of the agent's decoder. This may imply that objects are more important for decoding the current step than as a mean of maintaining a history of navigation.

Table 6.5. Ablation study results for EnvDrop model. COHS = Connectionwise Object Heading Similarity. OS = Object Summary.

Model	COHS	OS	SR (↑)
EnvDrop			39.4
	✓		**41.0**
	✓	✓	40.5

6.3. Qualitative Result

Figure 6.1 shows an example trajectory with its craft instruction for the base EnvDrop agent and our EnvDrop + OLC agent. Here, the base agent fails and ours succeeds. The base agent fails to exit the bedroom, whereas ours appears to have recognized the door and bedroom objects and associated them with the "exit the bedroom" action. This shows the relevance of understanding the objects it can see, so that it can follow an instruction that relies on these references.



(a) EnvDrop (fail)

(b) EnvDrop + OLC (success). The three objects with highest attention for the selected viewpoint are shown in a red bounding box.

Figure 6.1. Instruction: "Exit the bathroom to the closet walking by the left side of the pot. Walk forward. Go out of closet into the bedroom walking with the rack on your right. Make a left, go straight with the bed on your right. Exit the bedroom to the hallway. Make a right, exit the hallway to the bedroom." Example of an object focused trajectory from Ossandón et al. (2022) where the base agent (6.1(a)) fails and ours (6.1(b)) succeeds. The steps where trajectories diverge are surrounded in blue. The base agent fails to exit the bedroom (step 6), while ours recognize the door and bedroom objects, associating them with the "exit the bedroom" action.

7. CONCLUSIONS

The vision and language navigation task requires agents to reach a destination by following a natural language instruction. It is expected that, by giving agents the natural instructions and panoramic images as inputs and the ground truth paths as labels, they will be able to learn all the abilities required to complete this task. This work shows that, by giving the agents more detailed inputs and related tasks to learn, they will learn to navigate better, translating into improved performance in the relevant metrics.

We detail two methods for including object information into the agent's model architecture. The first is aimed for models that use a recurrent network as their base sequence-to-sequence architecture, and it's evaluated while integrated in the EnvDrop model. With this method, we learn that taking special care into properly contextualizing the objects that the agent sees is beneficial for action decoding, but not so much for maintaining a history of navigation.

The second method is for transformer based architectures, which is evaluated as an extension of HAMT model. With this, we learn that the benefits of object contextualization are not as high as with a recurrent architecture. This is mainly because the other pre-training tasks appear to already help in this regard.

Our main contribution is demonstrating that, by including object information during training, agents learn an improved navigation strategy. This translates into increased performance for the Room-to-Room task. Also, we see that, when instructions have a greater focus on using objects as references for navigation, they obtain an even better performance. This shows that objects are rich sources of information for this task.

Regardless of the obtained improvements, we also find that the performance increase is much greater in validation seen than validation unseen. This translates into a bigger generalization gap, showing that these models may be using these objects as a source of overfitting, which decreases the potential impact it has for unseen environments. More

care may be needed when training with this kind of data to avoid overfitting over seen environments. Section 7.1 proposes some ideas to help overcome this problem.

We hope that this work also shows the benefit of grounding language models with real world entities. We believe that visual information may help give concrete information of the abstract ideas that can be expressed through natural language.

7.1. Future Work

7.1.1. Train with all environments

Due to limited computing resources, the EnvDrop model was trained in a subset of Matterport3D environments. It is expected that the results obtained in this work should be the same when training on all environments, as the information given by our module aims to enrich the information received by the agent in these environments.

7.1.2. Use an object detector

Object metadata present in Matterport3D is messy and sometimes wrong. Some viewpoints don't have objects, and others report objects not present in that scene. Including an object detector pre-trained in indoor scenes for obtaining the object features and labels may improve their representations and, as a consequence, the agent's performance and generalization.

7.1.3. More auxiliary tasks

This work uses some auxiliary/pretraining tasks to help ground objects and rooms to their real world classification. These tasks are all supervised, but self-supervision usually has better generalization performance. Using self-supervised tasks may also help achieve a better, more generalizable, grounding. In particular, a task where the agent must predict

if the object is present in the instruction may help further the grounding of both textual and visual inputs of the model. This may result in increased performance and generalization.

7.1.4. Object data augmentation

To overcome the object overfitting problems shown in this work, an approach similar to Environmental Dropout may be taken with objects present in the scene. Objects seen by the agent may be replaced at the feature level with other objects of the same class, thus increasing the number of different types of the same object in each environment. With this, overfitting on particular variations of objects should decrease, and their visual representation will improve.

REFERENCES

Anderson, P., Wu, Q., Teney, D., Bruce, J., Johnson, M., Sünderhauf, N., ... van den Hengel, A. (2017). Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. *CoRR*, *abs/1711.07280*. Retrieved from http://arxiv.org/abs/1711.07280

Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Zitnick, C. L., & Parikh, D. (2015, December). Vqa: Visual question answering. In *Proceedings of the ieee international conference on computer vision (iccv)*.

Bengio, Y., Simard, P., & Frasconi, P. (1994, 02). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, *5*, 157-66. doi: 10.1109/72.279181

Biten, A. F., Tito, R., Mafla, A., Gomez, L., Rusiñol, M., Jawahar, C., ... Karatzas, D. (2019). Scene text visual question answering. In *2019 ieee/cvf international conference on computer vision (iccv)* (p. 4290-4300). doi: 10.1109/ICCV.2019.00439

Chang, A., Dai, A., Funkhouser, T., Halber, M., Niessner, M., Savva, M., ... Zhang, Y. (2017). Matterport3d: Learning from rgb-d data in indoor environments. *International Conference on 3D Vision (3DV)*.

Chen, S., Guhur, P.-L., Schmid, C., & Laptev, I. (2021). History aware multimodal transformer for vision-and-language navigation. In *Neurips*.

Chen, X., Fang, H., Lin, T.-Y., Vedantam, R., Gupta, S., Dollár, P., & Zitnick, C. L. (2015). Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019, June). BERT: Pre-training of

deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers)* (pp. 4171–4186). Minneapolis, Minnesota: Association for Computational Linguistics. Retrieved from https://aclanthology.org/N19-1423 doi: 10.18653/v1/N19-1423

Fried, D., Hu, R., Cirik, V., Rohrbach, A., Andreas, J., Morency, L., ... Darrell, T. (2018). Speaker-follower models for vision-and-language navigation. *CoRR*, *abs/1806.02724*. Retrieved from http://arxiv.org/abs/1806.02724

Gao, C., Chen, J., Liu, S., Wang, L., Zhang, Q., & Wu, Q. (2021, jun). Room-and-object aware knowledge reasoning for remote embodied referring expression. In *2021 ieee/cvf conference on computer vision and pattern recognition (cvpr)* (p. 3063-3072). Los Alamitos, CA, USA: IEEE Computer Society. Retrieved from https://doi.ieeecomputersociety.org/10.1109/CVPR46437.2021.00308 doi: 10.1109/CVPR46437.2021.00308

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep residual learning for image recognition. *CoRR*, *abs/1512.03385*. Retrieved from http://arxiv.org/abs/1512.03385

Hochreiter, S., & Schmidhuber, J. (1997, 12). Long short-term memory. *Neural computation*, *9*, 1735-80. doi: 10.1162/neco.1997.9.8.1735

Hong, Y., Wu, Q., Qi, Y., Opazo, C. R., & Gould, S. (2020). A recurrent vision-and-language BERT for navigation. *CoRR*, *abs/2011.13922*. Retrieved from https://arxiv.org/abs/2011.13922

Hu, R., Fried, D., Rohrbach, A., Klein, D., Darrell, T., & Saenko, K. (2019). Are you looking? grounding to multiple modalities in vision-and-language navigation. *CoRR*, *abs/1906.00347*. Retrieved from http://arxiv.org/abs/1906.00347

Jain, V., Magalhaes, G., Ku, A., Vaswani, A., Ie, E., & Baldridge, J. (2019, July). Stay

on the path: Instruction fidelity in vision-and-language navigation. In *Proceedings of the 57th annual meeting of the association for computational linguistics* (pp. 1862–1872). Florence, Italy: Association for Computational Linguistics. Retrieved from https://aclanthology.org/P19-1181 doi: 10.18653/v1/P19-1181

Kazemzadeh, S., Ordonez, V., Matten, M., & Berg, T. (2014). Referitgame: Referring to objects in photographs of natural scenes. In *Proceedings of the 2014 conference on empirical methods in natural language processing (emnlp)* (pp. 787–798).

Kolve, E., Mottaghi, R., Gordon, D., Zhu, Y., Gupta, A., & Farhadi, A. (2017). AI2-THOR: an interactive 3d environment for visual AI. *CoRR*, *abs/1712.05474*. Retrieved from http://arxiv.org/abs/1712.05474

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017, may). Imagenet classification with deep convolutional neural networks. *Commun. ACM*, *60*(6), 84–90. Retrieved from https://doi.org/10.1145/3065386 doi: 10.1145/3065386

Ku, A., Anderson, P., Patel, R., Ie, E., & Baldridge, J. (2020). Room-across-room: Multilingual vision-and-language navigation with dense spatiotemporal grounding. *CoRR*, *abs/2010.07954*. Retrieved from https://arxiv.org/abs/2010.07954

LeCun, Y., Bengio, Y., & Hinton, G. (2015, May). Deep learning. *Nature*, *521*(7553), 436–444. Retrieved from https://doi.org/10.1038/nature14539 doi: 10.1038/nature14539

Liu, C., Zhu, F., Chang, X., Liang, X., & Shen, Y. (2021). Vision-language navigation with random environmental mixup. *CoRR*, *abs/2106.07876*. Retrieved from https://arxiv.org/abs/2106.07876

Liu, Y., Zhuang, B., Shen, C., Chen, H., & Yin, W. (2019). Training compact neural networks via auxiliary overparameterization. *CoRR*, *abs/1909.02214*. Retrieved from http://arxiv.org/abs/1909.02214

Ma, C., Lu, J., Wu, Z., AlRegib, G., Kira, Z., Socher, R., & Xiong, C. (2019). Self-monitoring navigation agent via auxiliary progress estimation. *CoRR*, *abs/1901.03035*. Retrieved from http://arxiv.org/abs/1901.03035

Ma, C., Wu, Z., AlRegib, G., Xiong, C., & Kira, Z. (2019). The regretful agent: Heuristic-aided navigation through progress estimation. *CoRR*, *abs/1903.01602*. Retrieved from http://arxiv.org/abs/1903.01602

Majumdar, A., Shrivastava, A., Lee, S., Anderson, P., Parikh, D., & Batra, D. (2020). Improving vision-and-language navigation with image-text pairs from the web. *CoRR*, *abs/2004.14973*. Retrieved from https://arxiv.org/abs/2004.14973

Manterola, R. (2021). *Enhanced vision-language navigation by using scene recognition auxiliary task.*

Mao, J., Huang, J., Toshev, A., Camburu, O., Yuille, A., & Murphy, K. (2016). Generation and comprehension of unambiguous object descriptions. In *2016 ieee conference on computer vision and pattern recognition (cvpr)* (p. 11-20). doi: 10.1109/CVPR.2016.9

Mao, J., Xu, W., Yang, Y., Wang, J., Huang, Z., & Yuille, A. (2014). Deep captioning with multimodal recurrent neural networks (m-rnn). *arXiv preprint arXiv:1412.6632*.

Mezaris, V., Kompatsiaris, I., & Strintzis, M. (2003). An ontology approach to object-based image retrieval. In *Proceedings 2003 international conference on image processing (cat. no.03ch37429)* (Vol. 2, p. II-511). doi: 10.1109/ICIP.2003.1246729

Ossandón, J., Earle, B., & Soto, Á. (2022). Bridging the visual semantic gap in vln via semantically richer instructions. In S. Avidan, G. Brostow, M. Cissé, G. M. Farinella, & T. Hassner (Eds.), *Computer vision – eccv 2022* (pp. 54–69). Cham: Springer Nature Switzerland.

Qi, Y., Pan, Z., Hong, Y., Yang, M., van den Hengel, A., & Wu, Q. (2021). Know

what and know where: An object-and-room informed sequential BERT for indoor vision-language navigation. *CoRR*, *abs/2104.04167*. Retrieved from https://arxiv.org/abs/2104.04167

Qi, Y., Wu, Q., Anderson, P., Liu, M., Shen, C., & van den Hengel, A. (2019). RERERE: remote embodied referring expressions in real indoor environments. *CoRR*, *abs/1904.10151*. Retrieved from http://arxiv.org/abs/1904.10151

Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. *CoRR*, *abs/1804.02767*. Retrieved from http://arxiv.org/abs/1804.02767

Ren, S., He, K., Girshick, R. B., & Sun, J. (2015). Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, *abs/1506.01497*. Retrieved from http://arxiv.org/abs/1506.01497

Ruis, L., Andreas, J., Baroni, M., Bouchacourt, D., & Lake, B. M. (2020). A benchmark for systematic generalization in grounded language understanding. *CoRR*, *abs/2003.05161*. Retrieved from https://arxiv.org/abs/2003.05161

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, *115*(3), 211-252. doi: 10.1007/s11263-015-0816-y

Shen, S., Li, L. H., Tan, H., Bansal, M., Rohrbach, A., Chang, K., ... Keutzer, K. (2021). How much can CLIP benefit vision-and-language tasks? *CoRR*, *abs/2107.06383*. Retrieved from https://arxiv.org/abs/2107.06383

Shridhar, M., Thomason, J., Gordon, D., Bisk, Y., Han, W., Mottaghi, R., ... Fox, D. (2019). ALFRED: A benchmark for interpreting grounded instructions for everyday tasks. *CoRR*, *abs/1912.01734*. Retrieved from http://arxiv.org/abs/1912.01734

Singh, A., Natarjan, V., Shah, M., Jiang, Y., Chen, X., Parikh, D., & Rohrbach, M. (2019). Towards vqa models that can read. In *Proceedings of the ieee conference on computer*

vision and pattern recognition (p. 8317-8326).

Speer, R., Chin, J., & Havasi, C. (2016). Conceptnet 5.5: An open multilingual graph of general knowledge. *CoRR*, *abs/1612.03975*. Retrieved from http://arxiv.org/abs/1612.03975

Tan, H., & Bansal, M. (2019). LXMERT: learning cross-modality encoder representations from transformers. *CoRR*, *abs/1908.07490*. Retrieved from http://arxiv.org/abs/1908.07490

Tan, H., Yu, L., & Bansal, M. (2019). Learning to navigate unseen environments: Back translation with environmental dropout. *CoRR*, *abs/1904.04195*. Retrieved from http://arxiv.org/abs/1904.04195

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. *CoRR*, *abs/1706.03762*. Retrieved from http://arxiv.org/abs/1706.03762

Wang, H., Wang, W., Liang, W., Xiong, C., & Shen, J. (2021). Structured scene memory for vision-language navigation. *CoRR*, *abs/2103.03454*. Retrieved from https://arxiv.org/abs/2103.03454

Wang, H., Wang, W., Shu, T., Liang, W., & Shen, J. (2020). Active visual information gathering for vision-language navigation. *CoRR*, *abs/2007.08037*. Retrieved from https://arxiv.org/abs/2007.08037

Wu, H., Gao, Y., Guo, X., Al-Halah, Z., Rennie, S., Grauman, K., & Feris, R. (2021). Fashion iq: A new dataset towards retrieving images by natural language feedback. In *2021 ieee/cvf conference on computer vision and pattern recognition (cvpr)* (p. 11302-11312). doi: 10.1109/CVPR46437.2021.01115

Yatskar, M., Zettlemoyer, L., & Farhadi, A. (2016, June). Situation recognition: Visual semantic role labeling for image understanding. In *Proceedings of the ieee conference on*

computer vision and pattern recognition (cvpr).

Zhu, F., Zhu, Y., Chang, X., & Liang, X. (2019). Vision-language navigation with self-supervised auxiliary reasoning tasks. *CoRR*, *abs/1911.07883*. Retrieved from http://arxiv.org/abs/1911.07883

Zhu, F., Zhu, Y., Long, Y., Chang, X., & Liang, X. (2020). Vision language navigation with multi-granularity observation and auxiliary reasoning tasks..

Zhu, W., Hu, H., Chen, J., Deng, Z., Jain, V., Ie, E., & Sha, F. (2020, July). BabyWalk: Going farther in vision-and-language navigation by taking baby steps. In *Proceedings of the 58th annual meeting of the association for computational linguistics* (pp. 2539–2556). Online: Association for Computational Linguistics. Retrieved from https://aclanthology.org/2020.acl-main.229 doi: 10.18653/v1/2020.acl-main.229

Zhu, W., Qi, Y., Narayana, P., Sone, K., Basu, S., Wang, X. E., ... Wang, W. Y. (2021). Diagnosing vision-and-language navigation: What really matters. *CoRR*, *abs/2103.16561*. Retrieved from https://arxiv.org/abs/2103.16561

APPENDIX

A. DATA SOURCES

A.1. List of environments used during training and evaluation

For training the EnvDrop model with object data, a reduced set of environments were used because of resource limitations. A list of the used environments follows

Training: One third of all environments were selected at random.

Uxmj2M2itWa	82sE5b5pLXE	2n8kARJN3HM	1pXnuDYAj8r
VLzqgDo317F	p5wJjkQkbXX	r1Q1Z4BcV1o	HxpKQynjfin
PuKPg4mmafe	cV4RVeZvu5T	PX4nDJXEHrG	VFuaQ6m2Qom
JF19kD82Mey	sT4fr6TAbpF	E9uDoFAP3SH	XcA2TqTSSAj
8WUmhLawc2A	EDJbREhghzL	1LXtFkjw3qL	pRbA3pwrgk9
gZ6f7yhEvPG			

Validation Seen: The selected environments are the same as those used in training.

Validation Unseen: All original validation unseen environments were used.

A.2. Selection of objects to include in training

Object metadata in Matterport are classified into a set of classes called mpcat40.

All objects present in the selected environments were included, except those with *mpcat40* labels matching ”void”, ”wall”, ”floor”, ”ceiling” or ”unlabeled”.

B. CODEBASE

B.1. Code Sources

The code for our models was based on publicly available code on GitHub:

- Matterport3D: https://github.com/niessner/Matterport.
- Matterport3D simulator: https://github.com/peteanderson80/Matterport3DSimulator.
- EnvDrop model: https://github.com/airsplay/R2R-EnvDrop.
- HAMT model: https://github.com/cshizhe/VLN-HAMT.

B.2. Our code

The code for our models was based on publicly available code for Environmental Dropout and HAMT. Our editions of the code are available on the following forked repos:

- EnvDrop: https://github.com/MrEarle/R2R-EnvDrop-ObjAttn
- HAMT: https://github.com/MrEarle/HAMT-ObjAttn