UNIVERSITY OF GOTHENBURG

DEPARTMENT OF PHILOSOPHY, LINGUISTICS AND THEORY OF SCIENCE

EMBODIED QUESTION ANSWERING IN ROBOTIC ENVIRONMENT

Automatic generation of a synthetic question-answer data-set

Ali Aruqi

Master's Thesis:	30 credits
Programme:	Master's Programme in Language Technology
Level:	Advanced Level
Semester and year:	Autumn, 2021
Supervisor	Simon Dobnik, Nikolai llinykh
Examiner	Staffan Larsson
Report number	(number will be provided by the administrators)
Keywords	Embodied Question Answering, Question Generation, Spatial Relations, Synthetic Data-sets, Multi-Modality

Abstract

Embodied question answering is the task of asking a robot about objects in a 3D environment. The robot has to navigate the environment, find the entities in question, and then stop to answer the question. The answering system consists of navigation and visual-question-answering components. The agent is trained on a synthetic data-set of question-answers and navigational paths called EQA-MP3D. Each question in the data-set is an executable function that could be run in the environment to yield an answer. EQA-MP3D includes only two types of questions, color and location questions. The type of questions asked could be considered unnatural, and we observe that the question-answers contain biases.

Our work extends the data-set by automatically generating size and spatial questions. We generate a total of 19 207 question-answers for training and 3 186 question-answers for validation. Our data extension is intended to train the system to answer more question types and enhance the system's overall ability to perform the task.

Preface

I would like to thank all those who supported me throughout the work of this project.

Contents

1 Introduction

Imagine having a robot that could find the missing keys of the apartment and tell us that they are on the bed in the bedroom. Creating such a robot has been a central issue in artificial intelligence for a long time. In the ongoing advances in developing interactive and social robots, researchers aim at incorporating visual with linguistic understanding. Language, vision, and physical action are considered different modalities. A system that understands vision and language is, thus, a multi-modal system with multi-modal abilities.

Multi-modal abilities are essential for the robot's acquisition of more intelligent behavior. For the robot to be interactive, we would want it to understand language, have a vision, and perform physical actions. For successful actions such as finding the keys, the robot must understand the meaning of our question about the keys, use its vision to explore and navigate the surroundings, and identify the keys.

The ability to perform actions and interact in the physical world requires comprehension of the visual aspect of meaning. Our human ability to interact with our visual surroundings stems from faculties such as perception and memory (Regier, 1996). We conceive the physical world through perception. We make sense of what we see through a mental understanding of the perceptual input. In order to have an intelligent robot that performs complex tasks, as Russell & Norvig (1995) note, the robot has first to be able to understand and resolve references in its environment. Resolving references in an environment requires not only seeing but also understanding what is being seen.

Understanding what is being seen can be achieved by connecting the words' semantics with the perceptual input. As we conceive the world around us through perception, we express our conceptualization of the perceptual experience in words (Lakoff & Johnson, 2008). Therefore, comprehending the world around is necessitated by having a notion of meaning that associates 'words' with the visual/physical world (Nilsson, 2007).

Meaning concerning the world around us can be constructed through interactions in the world. The idea that meaning and intelligence emerge from interaction with the world is referred to as the *Embodiment Hypothesis* (Smith & Gasser, 2005). In embodied cognition, meaning is formed in a sensory-motor activity incorporating action and experience in the linguistic and visual world.

This introduction begins with discussing the notion of meaning. The discussion describes meaning with the different aspects contributing to the meaning formulation. In particular, the discussion focuses on how language, vision, and action are integral parts of creating a meaning construct of objects and words. The second part of the introduction reviews different methods of establishing meaning (grounding meaning) in artificial systems.

1.1 Meaning

The meaning of words is not a mere psychological phenomenon. Concrete nouns, for example, have references in the physical world, with physical properties indicated by their meaning. The meaning of a word is, thus, not only bound up with linguistic characters and mental notions but also with some physical representation in the world. For example, the word "chair" is represented by its token characters (c, h, a, i, r). The word also contains a perceptual symbolism(mental understanding/imagery of the chair's attributes and functions). At the same time, it refers to an entity with physical features in the world (Mooney, 2008).

In this triangular definition of meaning, vision has an integral part of the meaning. Vision represents the physical world to language. To recognize a chair, one should, for example, identify the existence of legs, seats, their sizes, and geometric shape in a visual scene. These properties of the physical reference of a chair can be represented in a visual form. Therefore visual recognition is part of the conceptualization process

that forms the perceptual symbol of an entity (Barsalou et al., 1999).

Perceptual information, however, is more than just visual information. The properties of an object include other sensory information such as the smell, taste, and texture of an object. For example, the meaning of rotten food could be more understood if the food is smelled or tasted.

The formation of a symbolic representation (meaning) requires more than the recognition of perceptual information. The construct of a meaning (symbol) could only be formulated by knowledge about the relations of the attributes that form an entity such as its color and shape. Barsalou et al. (1999) refers to the process of forming a symbol as 'componential' or schematic. "Meaning" in this view, is a scheme that is conceptualized or constructed. These schemes can be logically constructed in our mind by conveying truth about the world; or can be flawed conceptualization forming a false knowledge.

The whole meaning is the perceptual representation and the knowledge about it. The meaning of rotten apple is fully understood when we construct knowledge of the negative aspects of eating it. For example, rotten apples have red-brown colors. The stomachache that results from eating them leads us to believe that red-brown apples are different from all-red apples, not only by color but also other health/taste attributes, so we classify red-brown apples in a different category called "rotten apples". The knowledge about health implications and the attributes such as the colors and smell of a rotten apple help us categorize the rotten apple in the category of rotten food. Lakoff & Johnson (2008) explains that this attributive characterization can be expressed, for example, in the way we do prototyping and categorization of entities.

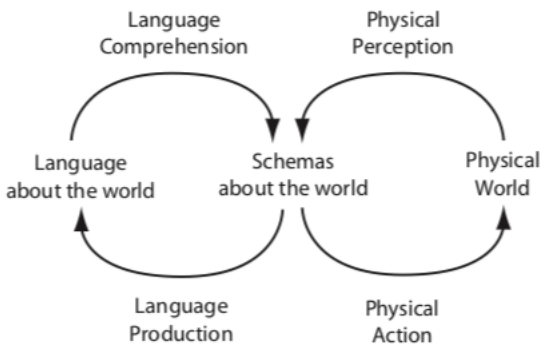


Figure 1: Meaning is formed with interaction in the world. The perceived meaning shapes our language and actions, and we express the formed meaning with language and action (Roy, 2005)

Interactions in the semantic world have an exchangeable nature. The interaction allows us to form meaning, and the formed meaning shapes our language and actions. In Figure 1 we see that the outcomes of our interaction in the world include not only linguistic implications but also affect the actions (Roy, 2005). 'schemes about the world' are the beliefs we make from the interactions. For example, our negative experience with the red-brown apple made us form the "belief" that rotten apples are bad. The knowledge that "rotten apples are bad" influences our future actions- makes us not eat the apples with "rotten" attribute.

Comprehending meaning through associating attributes with each other to form a belief or draw a conclusion denotes the notion of reasoning. The process of classifying the apple as rotten includes multiple abstractions. We might first identity the apple by its general shape structure. Then recognize, from previous experiences, that apples in red-brown color are not like all-red apples, then conclude that the apple is rotten. Reasoning is the ability to take the logical steps to conclude or make an inference.

1.2 Grounding Meaning

The approaches to grounding meaning (form meaning) vary depending on the aspect of meaning that each approach focuses on. The different methods we review below-approaching meaning as a mental notion, a map of connected knowledge nodes, vision and language representation, or the combination of different aspects of meaning representation.

Word-meaning in a Vector Semantic Space (VSM) can be described as representing a mental aspect of meaning (Turney & Pantel, 2010). Space can be understood by imagining our minds as a space that we allocate meaning representation in them. In VSM, the mind (represented as neural language model) is an artificial space. In this space, words meanings are allocated at different distances depending on their categorization, such as a rotten apple being closer to fresh apples than to a chair.

There are multiple hypotheses to representing word-meaning in a Vector Semantic Space (VSM). The Distributional Hypothesis is a popular example of word representation in VSM. The premise of this approach is that words with similar meaning tend to occur within the same context/text. The context, therefore, can define word meaning, such that the meaning of a word is represented by the words surrounding it (Turney & Pantel, 2010). The formulated word meaning representation is known as *word-embeddings* (Mikolov et al., 2013). Using language to define language has proven promising in inferential tasks, such as inferring that ”university” and ”student” are close to each other given their common context of ”education”.

On a related note, there are examples of research that attempt to incorporate knowledge graphs in the representation of meaning, such as Zhu et al. (2015, 2014).

Besides defining meaning in language, forming Visio-linguistic meaning is implemented using different methods. Early research in combining vision and language used probabilistic learning by aiming at drawing an alignment between sentences, phrases, and words with the corresponding perceptual representations (Lowe, 1999). An approach to probabilistic learning estimates the probability of a grammatical entity (text) being related to a perceptual representation (Zitnick et al., 2013). A second probabilistic method classifies each word in a sentence through the probability distribution of words over a perceptual representation. In Matuszek et al. (2012); Larsson (2015), we see examples of connecting entities of formal semantics with perception.

However, a widely used practice for combining language and vision is using multi-modal neural networks. Using neural networks for visual grounding is widespread across different multi-modal tasks such as VQA and Image Captioning. The basis of neural multi-modality generally aligns word embeddings with visual features in an encoder-decoder multi-modal architecture. In the following chapter, we will be reviewing and discussing the framework of the encode-decoder multi-modality.

2 Background

2.1 Multi-Modality

In this section we choose *Image Captioning* as an example of multi-modality with methods that give a general insight on language and vision tasks. In image captioning, as a vision-linguistic multi-modaility, vision and language are combined using feature extraction (Wang et al., 2020). Feature extraction methods can be used to process images, in combination with statistical model for language processing, as in (Fang et al., 2015; Zhang et al., 2005). However, a more common practice is to combine modalities by feature extraction for both language and vision, in an Encoder-Decoder mode.

2.1.1 Encoder-Decoder(CNN-RNN)

The Encoder-Decoder commonly consist of CNN and LSTM-RNN networks. Each of the networks is frequently used and modified in computer vision and language processing. In the following paragraphs, we briefly mention some tasks where CNN is used, then we show an illustration of its architecture and the factors contributing to its popularity. The text then moves to a short description of the LSTM/RNN networks. Finally, we illustrate the combination of the CNN-LSTM in an image captioning setup.

Convolutional neural networks are at the core of visual feature extraction, where CNN has a vital role in many computer vision tasks. We see CNN and its modified models (such as recurrent-CNN) used in tasks as object recognition (Liang & Hu, 2015; Girshick et al., 2016; Ren et al., 2015b), image classification (Simonyan & Zisserman, 2014; Krizhevsky et al., 2012), and semantic segmentation (Hariharan et al., 2015; Long et al., 2015).

A main reason for using CNN for image processing is its ability to reduce the high dimensionality of images. Image features contain large sizes represented in pixels which would require large number of parameters to train. CNN reduces the dimensions of an image by learning how to process a matrix from a large window such as 250x250 pixels into a smaller one as 25x25 or to a vector of 1X250 features. Through computing the convolution values of the image matrices and executing pooling computations, this process reduces the image into a smaller representation. The latter reduces the computational load and helps in processing and classifying the images faster.

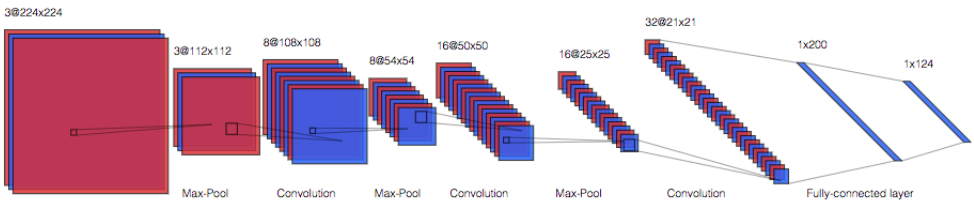


Figure 2: Example of a CNN image encoder.

In Figure 2 we see a dummy example of an image encoding where an image is encoded into a vector of features. The most left block resembles an image in pixels which consists of 3 channel (blocks), referred to on the top of the blocks preceding the @ sign, where each block is commonly representing one of three colors. Each of the three channels is better refereed to as a *feature map*. Convolutions produce a new feature map with reduced dimensions. The reduction occurs through calculating the dot product of filters with a matrix window from the feature map. Filters (k ernels) include learnt parameters that are updated simultaneously so that the value of the new map resulting from the dot product produce the best

representation of the features. In the example we see in Figure 2 the filter size is 5x5 and since there are three channels, the filter can be 3 dimensional so that the filter would be in size 5x5x3. The efficiency of this process is that the learnt wights for all the pixel-nodes would share the same filters (values) which in return saves computational load and time.

The max pooling throughout the encoder have a size of 2×2. In the second max pool block we see the dimensionality of each channel reduced by half as the max pool picks the max number from 2x2 windows in each matrix. The max number would be the most informative feature in the 2×2 window in the channel.

The number of filters in the convolutions blocks can be pre-defined and selected to be of any number. The more filters there is the more channels are produced. That's why we notice the number of channels (feature maps) increase in every convoltutional block. The dot product of the new filters with the input feature maps produce new channels- stacked in Figure 2. Finally the maps are passed to a dense layer (Fully connected layer) and reduced to a single dimension by calculating the dot product. The final output is a feature vector representing the image.

On the other hand, RNN's are known to be used widely in language technology applications. RNN is used, for example in text-to-speech (Arik et al., 2017), and machine-translation (Cho et al., 2014; Wu et al., 2016). The advantage that the RNN gives to these tasks is that the output size is not fixed and that each output depends on the previous one (The previous hidden layer). Such a sequence prediction is suitable for sentence predictions in respect to word dependency.

A frequent issue with RNNs is the vanishing gradient-descent. The gradient descent is an optimization algorithm that minimizes the error calculated in the loss function. Optimization, in brief description, is important for the learning process. It updates the model's parameters which determines the direction taken in the next time-step. This information is calculated given the input-output and the values of the parameters from the previous time-stamps. The gradients is reduced at every step due the value deductions in the activation function. When the gradient is reduced to almost zero value, it will be updating the parameters with no useful values, and therefore, learning ceases to improve.

LSTM, an extension of RNN, is usually selected as a modification to avoid the vanishing gradient. The architecture of the LSTM allows it to keep information stored for very long sequences. The latter enables the network to control the values of the gradient by updating it with information stored in the 'forget gate' from previous steps, possibly preventing the gradient from vanishing.

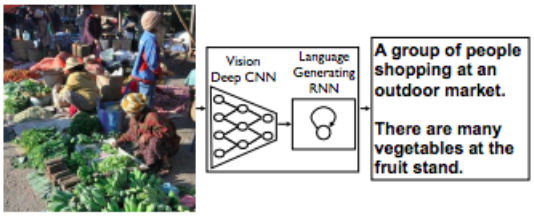


Figure 3: Example of a CNN-RNN image captioning model (Vinyals et al., 2015).

One of the first models of image captioning Vinyals et al. (2015) has encoder (CNN) and decoder (RNN). As seen in Figure 3 the neural model the CNN processes the image features, and the image feature vector is passed to an RNN to generate a description. This method is sequence modeling that is similar to machine translation. This means that image features are translated into words.

2.1.2 Attention Mechanisms

In humans, attention in vision and language occurs naturally. When we read a text to understand the main point we would be, for example, more focused on keywords and content words while reading. Psychology also explains that we have a mechanism of selective perception such that our attention gets focused on parts of the perceptual information that is sought after in our minds. Another example, if we look for oranges on a tree our vision would be fixated on the objects with orange color resembling the fruit with less attention to the rest of the tree.

Computationally, it has become a common practice to employ artificial attention mechanisms on the encoded input to boost a multi-modal ability to align visual and textual information. Learning attention in neural networks happen by learning to attend to specific regions of a visual input (Xu et al., 2016).

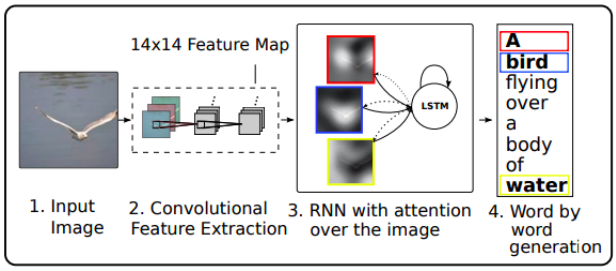


Figure 4: Image captioning with attention over the regions of the image (Xu et al., 2016).

When applying attention over the image, the image is presented by the feature map from the convultional blocks. As we seen in Figure 4, the regions are selected from the feature maps. The attention selects regions in order and predict the corresponded word. Other attention techniques encode the images already divided by regions instead of preforming attention over a feature map.

2.2 Dialogue and VQA

In this section, we discuss the capabilities of computers to exhibit more intelligent behaviour. Image-captioning and its methods showed an insight to how much computers could see and understand what its seeing. However, acquiring language in the visual world would require computers to be able to communicate what it sees. Otherwise, in order to say that a computer is visually or linguistically intelligent one should imagine the computer having to pass the Turing test in a visual surrounding.

Researchers attempt to improve systems that are capable to hold a dialogue with a visual content. Das et al. (2017) train a system of encoder-decoder model on a data set of interaction pairs with an image content.Skocaj et al. (2011) train a system on learning concepts with visual content in an interactive-learning approach. In the similar context of improving systems that are capable of having more natural interactions, we see example in Lin et al. (2014) of a VideoQA.

An essential element in the succession of an interactive visual dialogue with a robot is that the robot first understands the questions being asked to it within the visual context. Therefore, the learning of two shots interaction in VQA intuitively contributes to grounding in dialogue. However, this improvement can be seen as an improvement in one aspect of dialogue. The continuous and prolonged interactions in dialogue means that an overall improvement in dialogue are dependent on many other elements.

Antol et al. (2015) is the first notable data-set published for Visual Question answering (VQA). The data-set consist of open-ended and free-form questions. The data contains 250,207 images from MS COCO Lin et al.

(2015) and other abstract scenes. The question types in the dataset require a range of different capabilities such as common-sense reasoning, knowledge-based reasoning, object-detection and active recognition.

Data-sets that use MS COCO scenes such as Gao et al. (2015), Yu et al. (2015) in addition to Antol et al. (2015) used human workers to write the texts for the scenes. Other data-sets are generated automatically such as Ren et al. (2015a).

Zhu et al. (2016) introduce a unique QA data-set. The Visual7W consist of questions about an image with objects marked with regions in the image. Object grounding with image region introduced in Krishna et al. (2016) contains the largest data-set with regions for both VQA and Image-captioning. The object-region approach is intended to improve visual grounding, by marking the regions of the image that the strings refer to.

2.3 Embodied Question Answering

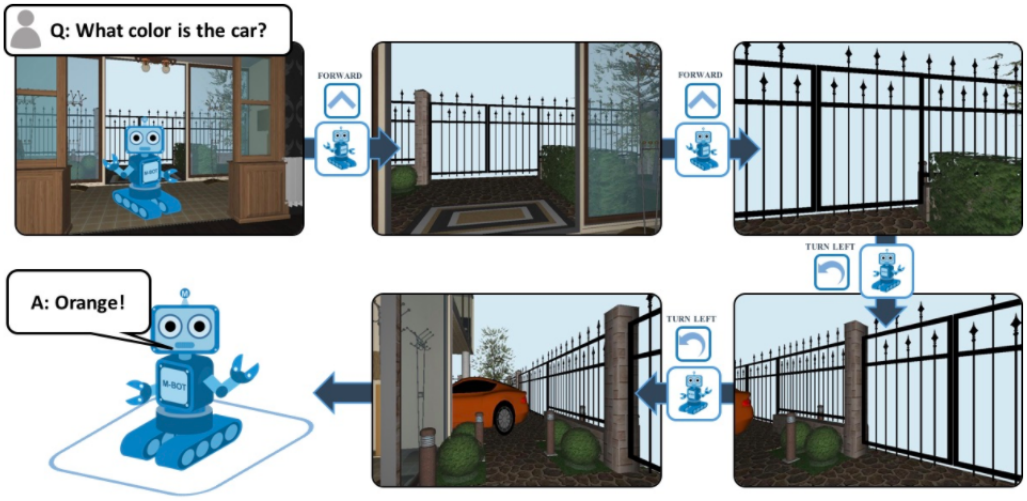


Figure 5: The Robot is asked a question at a start position. It needs to look around, collect information and decide on the next step to take. When it recognizes the car, it stops and processes the scene to answer the question (Wijmans et al., 2019).

Embodied Question Answering Das et al. (2018) [1] is a new interactive task presented as one of the tasks within the Habitat Platform (Savva et al., 2019; Szot et al., 2021). [2] The idea of the task is to allocate an agent at a random position in a 3D environment and ask it a question. To answer the question, the agent must intelligently explore the environment, collect information, and successfully navigate to the entity in question. EQA system navigates based on common reasoning, through an egocentric view, more or less imitating humans, it should be able to answer itself the common questions of ”where am I?”, ”where to go next?” and if asked a question about the car, as seen in Figure 5, it should be able to reason that cars are usually situated outside or in the garage and look for the exit. Once it navigates successfully to a point where it recognizes the car, the robot should stop and answer the question.

[1] Link to the official page of EQA. It also includes other published papers about the task `https://embodiedqa.org/`.

[2] Github link to the Habitat Platform. Information and code about EQA and the other tasks within Habitat can be found there `https://github.com/facebookresearch/habitat-lab`.

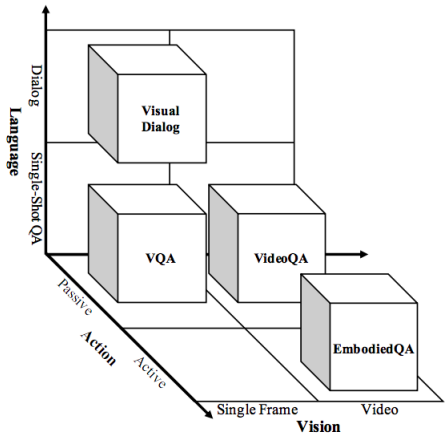


Figure 6: EQA in relation to other vision&language multi-modalities (Wijmans et al., 2019).

In Figure 6 we see where EQA stands relative to other vision-and-language multi-modalities discussed earlier. In the language domain, we see single-shot and dialogue. VQA is a typical example of a single shot interaction, where the system is designed to take a single shot question and a visual scene and output an answer. On the other side of the language domain (dialogue), we see Visual Dialogue, where the interaction within a visual context is continuous. On the vision domain, we see that VQA Visual dialogue is distinguished from VedioQA and EQA by the visual input type. The robot in EQA is continuously moving while navigating, so it inputs the vision similar to videos. Finally, EQA is distinguished from the rest of the multi-modalities on the action domain by being active. Hence, action here refers to executions of commands in a physical space. EQA is action-active by its navigational functionality. The rest of the modalities are passive with no functionalities of physical action execution.

The novelty of this system is that it presumably solves the problem of navigating and performing language tasks in unseen environments. Many of the earlier studies that deal with navigation, such as Kruijff et al. (2007); Lauria et al. (2001) require the system to have a localized map of the environment to be able to navigate in it. The problem of localization in robotic navigation is known as Simultaneous Localization and Map Building (SLAM) problem. SLAM is a problem where a robot should map an unknown environment without a GPS or local map. Simultaneous localization is when a robot discovers it is surrounding and simultaneously construct a map while aware of its changing location. This means that the robot should extract information from its surroundings and learn the map as it goes (Grisetti et al., 2010; Dissanayake et al., 2001; Zhang et al., 2018).

The answering system in the robot consists of two core components. The first is navigation, and the second is Visual Question Answering. In principle, the task should be performed in conjunction between the Nav and the VQA model. The navigation should lead the robot to a correct viewpoint then freeze its move. The VQA model should then take static image frames of the scene from the viewpoint where the Nav stopped and answer the question. However, the system's design allows it to exclusively perform either navigation or visual question answering on baseline models. The ability to train and evaluate either of the modules is possible due to two different training setups.

2.3.1 Training Setups

The first setup is a connected system with training in reinforcement learning setup. The training of the robot in RL happens based on the answer-based evaluation. The robot is rewarded if it completes the whole task using the two components connected. The basis of evaluation in the RL setup is the answer prediction. The system is rewarded if it answers the question correctly, and to answer the question correctly, it needs

to navigate to the right place and stop at a good view position so that the VQA system could have a relative and informative visual scene in order to answer the question. However, the researches in Das et al. (2018) elaborate that the system performs poorly when trained combined in RL. The navigation in the RL setup tends to position itself inaccurately at the stop-goal, which leads to passing distorted images to the VQA model. ”Noisy or absent views” would confuse the question-answering model (Das et al., 2018). For the mentioned reason, there is no available RL-based system available for developers.

The second setup is a system with the Nav and VQA components trained separately and differently. The navigation is trained in the 'Imitation Learning' setup, and VQA is trained in Supervised Learning. Hussein et al. (2017) describe Imitation Learning as learning with a teacher, where a robotic system has to mimic the steps taken by its tutor. Imitation Learning is considered an effective solution, in particular, for navigational problems as its step-to-step learning restricts the freedom of systems; we see IL popular, for example, in navigational systems of ground vehicles (Silver et al., 2008). The available Nav and VQA models that are available for training and evaluation in the habitat platform are the baseline models. The details of the training and the data used in each component will be described in more detail in the coming sections.

The answering system being researched in this project is the one **with navigation and VQA components trained and tested differently.**

2.3.2 Data

Figure 7: An example of an indoor 3D environment in MatterPort3D (Wijmans et al., 2019).

The dataset for the EQA task is called ”EQA-MP3D,” and it is a synthetic dataset generated automatically.[3] The EQA-MP3D task dataset is applicable for navigation and VQA, meaning that training the navigation and VQA use the same dataset. We refer to each question-answer in the EQA dataset as an **episode** because each QA sample includes a complete trainable navigation episode. We could describe the QA episode as a function executed in a 3D environment to yield an answer.

The 3D environments used in the task are indoor environments from the MatterPort 3D (MP3D) dataset Wijmans et al. (2019).[4]. The MatterPort3D is 3D constructed scene dataset which contains 90 segmented houses; Figure 7 is an example of 3D houses in MatterPort3D. In EQA the robot is trained in 57 MP3D environments and tested in 10 other unseen MatterPort3D environments.

[3]The dataset can be found on the Github page of the Habitat Platform, attached in the main page in the section' Task Data-sets' `https://github.com/facebookresearch/habitat-lab`.

[4]The GitHub reop of the MatterPort3D `https://github.com/niessner/Matterport`.

Data in Navigation Training In each QA episode, the information mainly used for navigation is a question, an ID for the 3D environment, a unique starting position, a destination goal, and a path to the destination. The mentioned navigational information, excluding environment ID and question, are all represented in coordinates. The starting position indicates where the agent should be spawned relative to the given environment. The path is the shortest path that the agent would take to reach the goal, consisting of steps and rotations. The shortest path is data used for Imitation Learning as the robots have to imitate the steps found in it. The goal is the stop point that marks the end of the episode. The stop point of the navigation is the viewpoint of the entity in question.

Data in VQA Training In each QA episode, the information used for training the VQA model is an ID for the 3D environment, a question, ground truth answer, and the position of view. The mentioned information is automatically taken in a code part of the Habitat platform and reconstructed into a conventional VQA dataset, QA pair, and a visual scene. The visual scenes are extracted using the view positions given in each QA episode in the corresponding 3D environment represented by the ID.

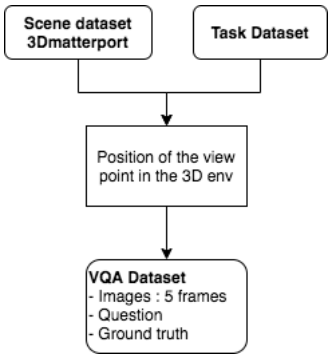


Figure 8: Locations of viewpoints of the entity taken from an EQA episode to extract a visual scene. The visual scene is then constructed with a QA pair to form a VQA sample of 5 frames of images, question and ground truth.

The extracted scenes for VQA consist of 5 frames images taken from the viewpoint where the navigator is supposed to stop. In Figure 8 we see an illustration of the structuring of the VQA dataset using the EQA task dataset (EQA-MP3D) and the scenes in MatterPort3D. The resulted VQA for VQA training is a Question-Answer pair with a visual scene.

Data-set Size & Question Types The question-answer data set contains three types of questions. Each question type is generated in a string template. The templates are as the following:

- **color_room** template: ”what color is <obj> in <room>?”In these questions, the agent needs to find the room in question, look for the object, and answer the question. For the agent to be successful at reaching its target, it needs to know the difference between rooms, and objects, by implicitly recognizing that a certain room is a living room and not a bathroom and such.
- **color** template: ”what color is <obj>”. The difference between *color* type and *color_room* is that no room is specified in the *color* type of question. In the *color* type, the agent needs to figure out where to look by itself. For example, ”what color is the fridge?” the robot needs to implicitly figure that the fridges are usually in the kitchen and navigate to the kitchen to answer the question. In other cases, the object could be in the vicinity of the robot’s starting point so that all it needs to do is to look around.

- **location** template: "What <room> is the <obj> located in". For location questions the robot has to find the object in question and recognize the room where the object is located.

In EQA-MP3D, each object in a question is unique in the room. The latter means that for an object to be selected for a question, there needs to be only one instance of that object existent in the room. The reason for this is to avoid ambiguity and not to confuse the agent if there happens to be more instances of the same object in the room.

There is a total of 11496 question episodes in the training split and 1950 question episodes in the validation set. As seen in Figure 9, in the training split, there are 1830 episodes of *color* type, 8031 episodes of *color_room* and "1635" of *location* type. For the validation split, there are 1335 *color_room* questions, 345 *color* questions, and 270 *location* questions.

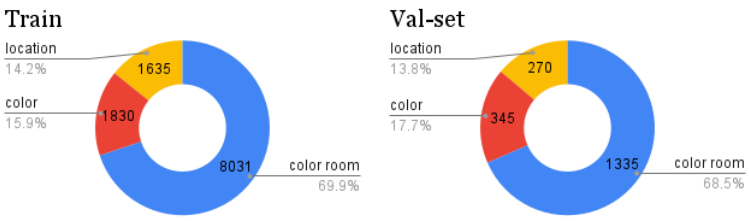


Figure 9: Number of question-answers represented by their types in the Train and Validation set.

However, the number of unique visual-question-answers is different from the number of episodes— a unique visual-question-answer is a question-answer with a unique visual scene and question-answer. For every unique question-answer and goal (scene), there are 15 different starting positions and shortest paths for the robot to train on for navigation. This means every unique QA in VQA is repeated 15 times. For example, in the validation set, the number of unique questions (QA and goal-scene) of *color* type is 23, we multiply it with 15 (the number of starting positions for every unique goal), and we get 345 episodes which is the number of episodes for *color* type in "Val-set" as seen in Figure 9. In the train set, the number of unique visual-question-answer for *color_room* is 536, for *color* is 122, for *location* is 109. In the validation set, the number of unique visual-question-answers for *color_room* is 89, for *color* is 23, for *location* is 18.

Data Bias In all color questions, (color & color_room), in the train set, we observe that 7% (41) of the unique visual-question-answers have a specific color as the only answer. For example, every time the question "what color is the picture in the hallway" is asked in a scene, the answer is always the same. In 20% (153) of the unique instances one color is the answer in 75% of the times, and in the remaining 25% times the answer is a specific second color. In 27% (181) of the unique instances the answer is always one of three colors, where one color in 51.2% of the times is the answer. For example the answer in the QA instances with the question "what color is the shelving in the hallway?" is always either *brown*, *slate grey* or *black* but brown is two times more likely to be the answer (51.2%). In the remaining instances one color has 17% more chance to be the answer over the other possible answers that were given for a particular question-string.[5]. Hence- The color annotations of the objects are assigned by human annotators, and the answers are generated automatically (Das et al., 2018).

In the question instances where the answer is always the same, the model would not train on disambiguation any color classes as it instead learns the answer to a particular question is always the same. In the remaining

[5]Link to the statistical analysis of the data in a notebook `https://github.com/Al-arug/EQA`.

questions, the model could possibly exploit the bias by learning that one color is more likely to be the answer to a particular question. In the evaluation section we will see examples of questions where the model's prediction to particular question is always a specific color, or predominantly a specific color.

2.3.3 Navigation Model

Habitat's navigation is referred to as PACMAN. It consists of two core components, planner and controller. The planner takes inputs from the vision and language model, and the encoding of hidden-layer and action of the previous time-step then outputs action-decision.

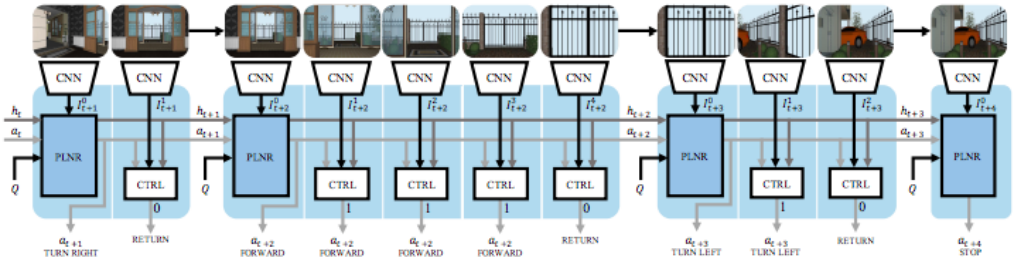


Figure 10: PLNR (Planner) analyses the visual and linguistic features and decides the next step to take. CTRL (Controler) executes the step (Wijmans et al., 2019).

The controller takes the previous hidden state and action decision and executes the action. As seen in Figure 10, visual input is passed to the control then the controller classifies the following decision of two possible decisions. The controller (CTRL) either repeats the last action given by the planner or returns the control to the planner. The controller can repeat the same action maximum of five times then it automatically returns to the planner.

Visualization of the navigation is in Figure 10. T stands for the planner's time-steps, t = 1,2,3...., and N(t), n = 0,1,2,3.. denotes the controllers time-steps. The denotations of symbols explained clearer in the quotation below:

"I_t^n denotes the encoding of the observed image at t-th planner-time and n-th controller-time. The planner is instantiated as an LSTM. Thus, it maintains a hidden state h^t (updated only at planner timesteps), and samples action $a_t \in \{forward,\ turn-left,\ turn-right,\ stop\}$" (Wijmans et al., 2019) p(6).

For example, the first step-decision from the planner is denoted as such:

$$a_t, h_t \leftarrow PLNR\,(h_{t-1}, I_t^o, Q, a_{t-1})$$

The planner computes the next step-action a_{t+1} from input of the previous hidden layer (h_{t-1}), question encoding (Q), the previous action a_{t-1}, and the image input given to the PlNR (tI_t^o). The planner selects the action a_{t+1} and update the hidden state h_{t+1} then passes the control to the controller.

The controller decides to either repeat the action or return control to the planner. The controller's classification is based on the current hidden-stateh_t and current action a_t and the image observation from the planner + the image given at the controller's time-step. The denotation of the classification is as such:

$$\{0,1\} \ni \; c_n^t \leftarrow CTRL\left(h_t, a_t, I_t^n\right)$$

If $c_n^t = 1$ then the action a_t repeats. Else $c_n^t = 0$ or a max of 5 controller-times been reached, control is returned to the planner (Wijmans et al., 2019)p(6). The h_t a_t coming from the planner act as an intent. The controller, initiated as "feed-forward multi-layer perceptron with 1 hidden layer" Wijmans et al. (2019), repeats and controls the action in order to align I_t^n with intent given by the planner.

2.3.4 VQA Model

The VQA model is a CNN-LSTM architecture. The CNN encodes 224x224 RGB images with a "multi-task pixel-to-pixel prediction framework" (p6) encoding. The structure of the CNN4 5x5 Conv, BatchNorm, ReLU, 2x2 Max-Pool blocks, and they produce a fixed-size representation. "The range of depth values for every pixel lies in the range r0, 1s, and the segmentation is done over 191 classes" (Das et al., 2018)(p.11). The "lstm" is a 2-layer LSTM with 128d hidden layers.

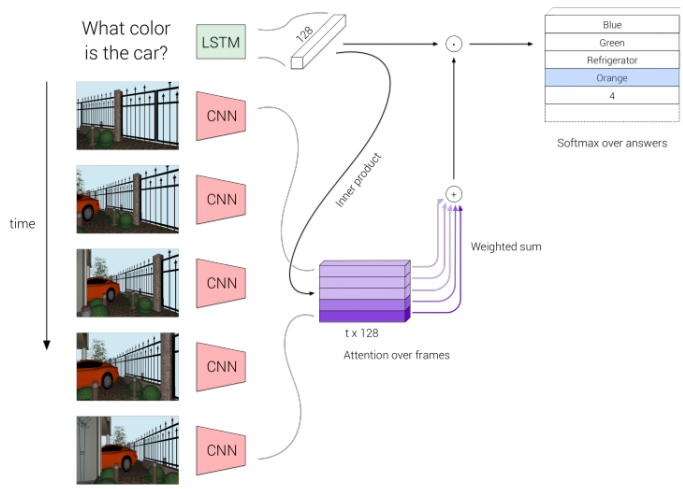


Figure 11: Architecture of the VQA model consist of an LSTM for language encoding, and CNN for vision. One of the five imag frames is selected through attention (Wijmans et al., 2019).

First, the CNN extracts features from five images (5 frames) scene, and the LSTM encodes the textual features of the question. Second, dot product attention is preformed over the words and each of the frames. Third, a softmax converts the question and image similarity into attention weights then the question encoding is concatenated with them (Das et al., 2018)(p6). Fourth, the concatenated features are classified in a softmax, where the answer probability is distributed over thirty five answers.

2.4 Problem

In an experiment we conducted on the VQA model in Das et al. (2018), we observed that the system tends to answer the questions relying mainly on the textual input in the questions (bias)[6]. The idea of the experiment was to give the model a random image instead of the original scene and see if it affects its predictions. The results showed that the system gave correct answers despite the absence of the corresponding scene

[6]Link to the experiment "Testing VQA's reliance on vision "`https://github.com/Al-arug/Habitat-Project`

required to answer the question. In such a case, the system's performance would typically have worsened, not improved, as the required visual information to answer the question is missing. The correct answering by the system was demonstrated in an overall increase in the performance score. Its ability to answer correctly demonstrates its reliance on the language model to predict the answer.

The system's ability to predict answers correctly in the experiment indicates a lack of visual grounding. We draw this conclusion from the observation that vision did not influence the predictions. This means that the system, in training, has not learned a scheme for word-meaning in association with vision. Grounding language in vision is when we connect the "high-level" symbolic representations such as language to a "low-level" non-symbolic representation such as the sensory (visual) features. The ability to ground language in vision is essential for any task requiring "seeing" and attending answer. If a robot successfully learns to align and combine the two types of representations, one could say that the computer understands what it sees (visual grounding). When a system fails to achieve such a connection, we define the problem as the "Symbol grounding problem" Harnad (1990) or "lack of visual grounding"

We presume that the lack of visual grounding is attributed to bias in the dataset. Earlier in this text, we reviewed textual biases within the EQA dataset. Having biases in the dataset would hinder the learning process, as it gives the model a way to learn to avoid combining vision and language by giving correct answers without actually learning to combine the two types of data.

We also observe that the type of questions asked are simplistic and can be considered unnatural. The existent color questions in the dataset are not the type of questions that a human would naturally ask. The limited types of questions found in the dataset seem to be meant to simplify the robot's task with a primary focus on navigation.

2.5 Problem In a Context

Selvaraju et al. (2020); Goyal et al. (2017)) and other research within the VQA point out the problem where models learn biases in training and manage to give good results in the testing. Johnson et al. (2017) elaborate that the underlying issue here is that the model answers by memorizing prior textual information. For example, a neural network might answer the question "What covers the ground?" correctly by answering "snow," "not because it understands the scene but because biased datasets often ask questions about the ground when it is snow-covered" (Johnson et al., 2017). Fukui et al. (2016) clarify that the models' answer-cheating is demonstrated when a VQA system primarily relies on the language model and ignores the visual information. Such a learning problem is crucial because it makes it challenging to evaluate the model's improvements (Agrawal et al., 2018).

When a system cheats its way into answering the questions, it shows a lack of visual grounding (Goyal et al., 2017). Visual grounding (understanding the meaning of words about vision) is crucial because we want the systems to understand the reasoning steps that humans would logically take to answer a question (Agrawal et al., 2016; Zhang et al., 2016; Fukui et al., 2016). For the system to be able to reason its way to predict an answer, it must first capture the full meaning. Selvaraju et al. (2020) explain that learning to reason would require the systems to make inferences at "multiple levels of abstraction." For example, "is the banana ripe?" where it would instantly answer "no." Answering this question would require the system to rely on perception to answer sub-questions such as where is the object? What are its shape, size, and color? Then reason that the "yellow" color indicates ripeness (Selvaraju et al., 2020).

2.6 Research Questions

- How can we extend the dataset with more sophisticated and natural questions? (A useful robot should answer a variety of questions.)

  Adding new questions could help test the system's capabilities, but more importantly, we consider it a step to enhance the system's cognition. The VQA system that we are improving is part of a robotic system that should ideally be helpful for human use. A social robot's usability is very dependent on its exhibition of human-like intelligence (Fong et al., 2003).

- How does the VQA system perform with the new question types?

- Does asking questions of spatial and size types improve the system's attention to vision? (Evaluating it based on the performance on color questions)

3 Methods and Materials

3.1 Overview

This section describes the methods and materials sources we use for question generation and the training and evaluation of the EQA system. The primary method relies on using utilities provided by the habitat platform. The name 'Habitat' is derived from the notion of learning within and from an environment (Savva et al., 2019). The utilities in the platform provide necessary arrangements for the EQA task, such as simulating and working in a 3D environment, spawning a robot with a specific configuration, preparing data sets, and setting up models for training. Materials are mainly used for question generation. The materials consist of data sets that contain semantic annotations and other geometric information essential for generating trainable episodes for navigation and VQA.

The two components providing utilities in the Habitat Platform are referred to as 'Habitat-sim'[7], and 'Habitat-Lab'[8]. Habitat Simulator is a 3D simulator with multiple functionalities, such as facilitating configurable sensors and robots in 3D environments. The habitat lab is a library that contains multiple tasks that can be performed in the environment. The lab provides different models and training setups. In the following sections, we describe the two components and their usage in this project.

The material we use for question generation is extracted from EQA-MP3D and MatterPort3D. The environments in MatterPort3D contain semantic annotations necessary for generating questions and the general linguistic understanding of the space. In addition, the annotations come with geometric information about the entities in the house, such as coordinate locations. The geometric information is essential for understanding the space geometrically for navigation as well as for question making. For example, asking a question about a spatial relation between two objects requires knowing their location on a global map. The task dataset, EQA-MP3D, provides data for navigational training and information about the objects in the questions. Following the Habitat Lab and Simulator description, we outline the relevant material in the two data sets and explain some of the concepts necessary for understanding the usability of the extracted data.

3.2 Habitat Simulator

Habitat-sim simulates 3D environments assimilating real-world settings. The environments are based on constructing either synthetic or real-world based scenes. Szot et al. (2021) describes a simulator as a system of two parts, physics engine, and renderer. The physics engine generates physical phenomena such as gravity and the physical state of the environment throughout the simulation, and the renderer completes the physics engine's work by outlining objects, colors, and borders. When constructing the scene, habitat-sim can do environment state manipulation by changing the layout of objects (Savva et al., 2019). We observe a change of object's layout, for example, in the names of objects in MatterPort3D extracted from habitat-sim and the ones found in the MP3D annotation files.

Habitat-sim has efficient GPU usage and can simulate different environment data sets. The simulation is displayed on GPU devices, which usually would require big storage of GPU to display a simulation of houses. The simulation setups in Szot et al. (2021); Savva et al. (2019) allow for smaller storage GPU's to perform the simulation. The latter makes it possible for an unfamiliar user of 3D simulation to simulate on their machines with reasonable speed. For the data set part, the simulator is designed with generalization for simulating different 3D Detests. In addition to MP3D, it supports 3D simulating for the following 3D datasets: GIBSON Xia et al. (2018), Replica Straub et al. (2019).

[7]The GitHub repo for Habitat simulator `https://github.com/facebookresearch/habitat-sim`
[8]The GitHub repo for Habitat Lab `https://github.com/Al-arug/habitat-lab`

Habitat simulator facilitates the employment of configurable sensors and agents. Configurations such as the location to spawn the agent and the type of sensors, and their position on the agent are the types of flexible settings given to the simulator to act upon. "Sensors" is a different name for referring to the CNN decoders where each decoder can be seen as a sensor of the following: 1) RGB reconstruction, 2) semantic segmentation, and 3) depth estimation. The latter sensors are used to obtain "object attributes (i.e., colors and textures), semantics (i.e., object categories), and environmental geometry (i.e., depth).". The agent can be configured with or more of the mentioned sensors depending on the task. For navigation, the agent is configured with "depth" and "RGB" sensors. A depth sensor is essential for the agent's capability to navigate. With a depth sensor, it could estimate distances and avoid colliding with obstacles. For VQA, the agent is configured to output only "RGB" images of 5 frames.

3.3 Habitat Lab

Habitat lab can be described as an API that facilitates task training in connection with the 3D simulator. In addition to initiating the simulator, Habitat-lab provides trainers, neural models, and data loaders. The lab has a hierarchical structure that acts given different configurations. For example, the steps for training/evaluating a task would be to prepare the data, initiate models and trainer, then simulate 3D environments. The information about the required model and the data for each task, such as paths to data and the models to use, can be manually given/changed in a task's configuration file.

In addition to facilitating the training of whole tasks, the lab can train models separately. Concerning EQA, the CNN model, in particular, is trainable independent of the other components using the lab platform.

Training and evaluating the navigation and VQA is possible on baseline models[9]. The text-image attention model is trained in connection with the pre-trained CNN[10] [11]. For navigation, the platform provides a training setup for Imitation Learning.

As mentioned earlier, for a VQA training and evaluation session, the lab facilitates preparing the VQA dataset in a conventional VQA format. Data preparation and setting up the models & trainer acts upon the paths and settings given in the VQA configuration file [12]. The configuration file also includes some manually instructed commands for configuring the robot. The lab passes these instructions to the simulator. The simulator is initiated simultaneously while preparing the VQA dataset. The data loader of the lab takes each environment ID from each QA sample in the EQA dataset and extracts the image frames.

Besides facilitating EQA training, the lab contains training setups for other tasks. The trainable tasks in the platform are as the following Goal navigation where the robot has to navigate to a geometric point; object navigation where the system has to navigate to an object; pick-up task where the robot has to pick up an object and move it to a different location, language-vision task where the agent follows directional instructions. Each of the previously mentioned tasks has a separate data set.

[9] Instructions for training and evaluating the baseline models in the API `https://github.com/facebookresearch/habitat-lab/tree/master/habitat_baselines/il`

[10] File containing the VQA trainer `https://github.com/facebookresearch/habitat-lab/blob/master/habitat_baselines/il/trainers/vqa_trainer.py`

[11] File containing the VQA model `https://github.com/facebookresearch/habitat-lab/blob/master/habitat_baselines/il/models/models.py`

[12] Configiration files for baseline including VQA config `https://github.com/facebookresearch/habitat-lab/tree/master/habitat_baselines/config/eqa`

3.4 Data and Data-sets

In this section, we review the materials used in generating QA episodes. The first part is a review of data structure and the relative semantic annotations found in MatterPort3D. The review of MP3D also includes an elaboration of geometric and viewpoints concepts necessary to understand the geometric annotations in MP3D and their usage. The second part of this section includes a review of the EQA-MP3D dataset structure and content.

EQA-MP3D is seen as a method and material source. The method of generating QA episodes relies on imitating EQA-MP3D. The imitation of EQA-MP3D ensures that the newly generated QA episodes are executable in the habitat platform by matching code requirements. Having the same structure for the generated question-answers as EQA-MP3D, a review on EQA-MP3D would also help understand the shape and form of question-answers generated in this project achieves. The navigational data we take as a material source for navigational training is highlighted and explained in the same review.

3.4.1 Semantic Annotations in MatterPort3D

Annotations In the MatterPort3D annotations, each house environment has four files. The four files are x.house, x.ply, x.glp, and x.navmesh. We collect the annotations from the x.house files house.

Each house file (x.house) has eleven line-types of annotations.. The lines are marked by a capital letter as a marker; the first letter-marking to the last letter areas in this list [H, L, R, P, S, V, P, I, C, O, V]. Each letter-marker symbolizes a certain type of information. In this section, I will explain only the type of information that we use in this project.

The line representing an object's info in a house file begins with the string "O". The "O" lines contain information about the objects in the house. Every line that begins with an O letter consists of one object in the house with corresponding information about its geometry and location within a room and level floor. Each "O" line looks as such: [O object_index region_index category_index px py pz a0x a0y a0z a1x a1y a1z r0 r1 r2 0 0 0 0 0 0 0 0 0]

The object's data in the line seen above comes in a string form. Each section in the string represents different types of information. *Object_index*, the index of an object is what we refer to as the object ID. *region index* is the room ID. *category_index* is the object's index in the category map; this index is used to obtain the object's name from the category map.*px py pz* represent the center of the box in (x,y,z) axis. *r0 r1 r2* represent the radius of the object from the center on the (x,y,z).*a0x a0y a0z a1x a1y a1z* these are the rotation of the OOBB radius(OOBB and radius will be elaborated on in a coming section). Finally, the last "0" s in the line have no meaningful value and therefore are ignored.

Points of View, Geometric Data As seen in the previous section, the geometric information consists of elements as the location of an object, region, or level, defined by their center in a world coordinate system. Other information is the size of the entity given its radius from its starting location (center).

The camera views of the scenes are globally oriented (Chang et al., 2017)(p3). A way to allocate an object is to find its location under global coordinates. The global coordinates start from the center of a house where the center of the house is (0,0,0) on the (x,y,z). Moreover, let us say all the objects are positioned throughout the house's (x,y,z) axis where its distance defines the location of each object to the house center.

[12]https://github.com/niessner/Matterport/blob/master/data_organization.md

When annotated, the objects are viewed through a camera. The description of their geometric location, thus, should consider the view-position of the camera.

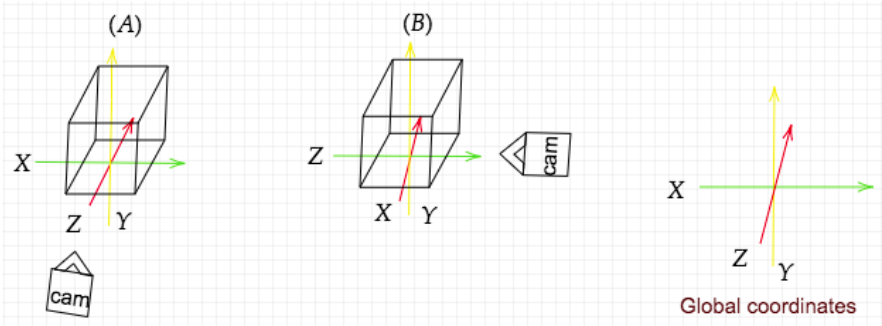


Figure 12: The camera in graph A views the objects from global perspective(readers perspective). The view of the camera in graph B is rotated. The rotation is resembled in the the axis's representation.

In graph (A) in Figure 12, we see that the camera-view of coordinates aligns with the global coordinates. The (x,y,z) that go through each object in graph (A) and graph (B) are the view of the axis in reference to the camera. However, suppose the camera is positioned to the right of the object from our view, as in graph (B). In that case, we say that the camera view of coordinates is not aligned with the global view. We notice in the graph (B) that from the camera view, the "global X" is "Y" and vice versa.

Some geometric calculations cannot be performed if the location measurements are not relative to each other. For example, if we want to calculate the distance between objects, the locations must be consistent with one reference point. The camera position is changing, and if the camera's position references the location of an object, we would get locations relative to the changing position of the camera in the timespan.

To globalize the view's orientation, one could use measures such as top-down view of a map or calculating the camera's rotation from the global center. While the global locations are crucial for measuring the distance, other point-views are also crucial for other purposes. There are three essential coordinate systems to know when working in a 3D environment:

1. World coordinates (global): World coordinates(global): The coordinate system that starts at the center of the world; a house in our example. The object's center is located by its distance from the center of the world.

2. Camera-view coordinates: The coordinates from the camera's views. The center of this coordinate system is the position of the camera. Its distance to the camera defines the center of the object in this world.

3. Local view: The center of the local view is the object itself.

The center of all these views to themselves is (0,0,0). We described above that the world coordinate system allows us to measure the distance between objects in a world map. The camera view helps provide precise geometric views of what a robot is seeing while moving. The local view could tell about the size of an object. In particular, the (x,y,z) from a local point of view tells how far the object stretches from its center where the center is (0,0,0) in the local view.

MatterPort3D provides the views described above. Next, We discuss the usage of the object's location in global coordinates and the local view in detail.

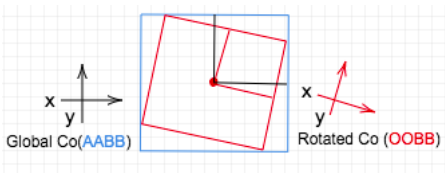


Figure 13: 2D AABB represented in blue square with its axis aligning with the world view of axis. 2D OOBB represented in red square has its axis rotated from the global view.

Object's Locations, Geometric Data In a 3D environment, objects could be represented by bounding boxes(covers) referred to as "Axis-Aligned Bounding Box" (AABB) and "Object-Oriented Bounding Box" (OOBB). The AABB can be described as being oriented with a global view, and OOBB is oriented with a local view.

The AABB and its center are aligned with the view of the world coordinates, while the OOBB is oriented with the box and more likely to be rotated from the global view. In Figure 13 we see a demonstration of the two boxes in 2d squares. The red square represents the 'Object-Oriented Bounding Box' (OOBB), where its (x,y) radius connecting the center to the sides is colored in red. The blue square represents the 'Axis Aligned Bounding box' (AABB), where its (x,y) radius connecting the center to the sides is colored in blue. The notable difference between the two boxes is the direction of their radius. The radius of the AABB in black is aligned with the direction of the global coordinates on the right part in Figure 13. The radius of OOBB, on the other hand, has its coordinate direction rotated from the global coordinates illustrated in the red coordinates on the right side of the graph.

The difference in the rotation of the coordinates is important for determining the correct calculation for estimating the locations of the box's sides. In order to obtain where the sides/corners of the box are located in the global coordinates, we would estimate how far the radius-es stretch from the center and in what direction.

The estimation for the AABB sides is straightforward since the AABB's radius stretches in the same direction as the global coordinates. For example, subtracting the length of the AABB radius on the y-axis from the center would give us a location point of the lower horizontal line of the blue box. Adding the (y) length from the center point would give a location point on the upper side of the blue box. Hence, the center and the AABB radius are globally aligned(pointing in the same direction). The latter implies that adding or subtracting them would give the correct globally defined position of the AABB side.

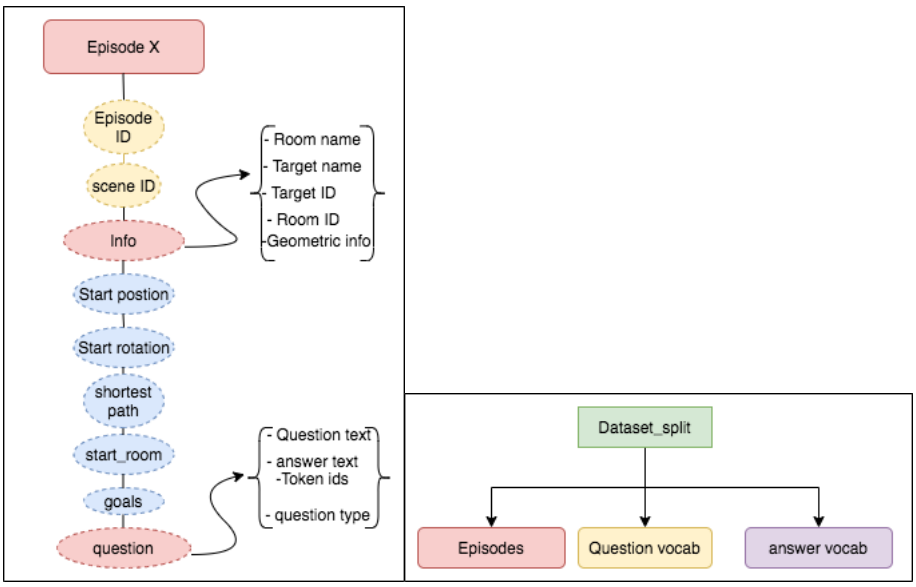
Even though the centers of OOBB and AABB are positioned in the same location, the direction of their axis is different. For estimating the sides of the OOBB from the center, one should consider the rotation of the coordinates. Adding or subtracting the OOBB (x,y) radius from the center, as done for the AABB in the previous example, would likely not end in a correct position on any OOBB sides. The latter occurs because the directions (slope) of the OOBB radius and the global coordinates are different. Estimating a globally defined point on any OOBB sides would require calculating the rotation or the slop of the radius from the center (the direction).

Using the AABB of an object is suitable for a direct allocation of the geometric locations of objects. At the same time, OOBB could give a more accurate estimation of the size of an object. With AABB, the borders of an object could be found with straightforward calculation, which makes it less complex. For example, allocating two objects and calculating the distance between any side would only require knowing their radius and centers. The OOBB, on the other hand, is more enclosed in the object since it is more oriented in the object's local view. The enclosing of the OOBB on the object makes the space between the box sides and the object's edges much smaller compared to the space that could be found in an object

defined in an AABB. The radius of the OOBB can provide a more accurate measurement of the object's volume. For size estimation, the rotation is unnecessary because calculating the volume does not require knowing any coordinate positions in a global map.

3.4.2 EQA (Task Dataset)

Our method for generating questions relies on imitating the structure of the EQA-MP3D dataset. In this section, we give a review of the EQA-MP3D structure and content. The EQA-MP3D also provides a primary material source. The relevant material consists of the navigational data required for constructing a trainable episode. In addition, an episode includes essential information about the target object in a question, such as its ID and room ID. This object's info is important as it would make it possible to access more metadata about the specific object from the annotation.



(a) (b)

Figure 14: (a) represents the structure of one QA episode. The arrow and curly brackets branching from "info" and "questions" show the content of each of these two categories. (b) Represents the most top layer of a split("Train" or "Validation").

Structure .

Figure 14 (b) shows the top structure of the validation and train sets. *Episodes* contains all QA episodes in a data-set split. *Question vocab* and *answer vocab* contain the same elements as dictionary keys. The elements are: [word list, stoi, itos, num vocab, pad token]. Word list, is all the words used in the dataset, "stoi" denotes string-to-integer map, and "itos" denotes integer-to-string map. "num vocab" is the number of vocabulary. "Question vocab" and "answer vocab" in the "train" and "validation" sets are identical to each other.

Each question-answer pair is an episode that consists of multiple layer information. The structure of an episode is as seen in Figure 14 (a). We describe the elements of an episode in the following:

- **Scene ID**: Scene ID is the the house ID given by the house ids in MatterPort3D.

- **Episode ID**: Episode ID is the episode's index in the data-set.

- **Info**: This element contains all the information about the object and room in a question. The inner layers of "info" include the following:

  Information about the target object is the first layer within **Info** and its elements are listed below:

  - *Centroid*: The center of the object's Axis-aligned bounding box(AABB). The center coordinates are in 3D on the (x,y,z) axis and defined with the global view.
  - *Radi*: This is the radius of the Axis-Aligned-bounding box of the object. The AABB radius is in 3D on the (x,y,z) axis.
  - *Level*: The level-floor number in the house where the object is located.
  - *Room ID*, *Object ID*,*Room name*, and *Object name* : Room ID, room name and object ID as given by semantic annotation in MatterPort3D. Many of the objects are re-named, mostly names in hyponymes changed to hypernym category such as: round-sofa, l-shaped sofa changed to their hypernym category "sofa". The information about the room (*Room ID*,and *Room name*) are found in the second layer within "info"

  The elements that are marked in blue in Figure 14 are *navigation data* used for training the navigator:

- **Start position**: The start positions are all unique. For each unique question in the data set, there is fifteen different starting position.

- **Rotations**: These are the rotations that the agent has to do while navigating. It stands as supplementary information for the shortest path.

- **Shortest Path**: The shortest path is the data used for Imitation Learning by training the robot to follow the steps in the path with short rewards. The path consists of steps in frequencies that mark the shortest way to reach the question's object.

- **Goals**: Goals are the destinations that the agent should reach in navigation. The goals stand for the viewpoint where the robot can see the target object. Each viewpoint consists of a geometric position and the rotation toward the target object respective to the position.

In the following chapter we elaborate on the process of generating questions and episodes. The generated episodes in a data-set have the same structure as the episode described above.

4 Task One - Question and Answer Generation

4.1 Overview

This section presents the process of generating question-answers. In this overview we first state the type of questions generated and the motivation behind the selection of these types. Second, the overview presents a general idea of how the generation is done. Third, we list the general structure of the work and the report.

4.1.1 What Questions are Generated and Why?

We choose spatial and size questions for the importance of the ability to answer such questions in terms of the robot's practicality and exhibition of more intelligent behaviour. In language we express much of our thoughts, which are non-physical constructs, using spatial relations (Lakoff & Johnson, 2008); we say for example that an idea was at the *top of my mind* or that one has *big ideas*.

Furthermore, Barsalou et al. (1999)(p.616) notes that the role of spatial relations is central to knowledge processing and to cognition in general. Neural representation including perceptual ones are spatially organized in our cognition (Barsalou et al., 1999); For example, to describe the latter in an analogy, we perceive perceptional sensory-motor information in *top-down* or *bottom-up* processing. The size of objects or its shape is also an integral part of identifying spatial relation between a pair of objects (Dobnik, 2009).

In terms of practicality, spatial reasoning and spatial language are natural and very essential for many practical tasks. A robot would be very helpful if it could look up if the phone is *on* the table or knows to bring the right spoon when asked to bring a small spoon.

Our choice of size and spatial questions also considers the aspect were the questions would contribute to better visual grounding. Understanding the word meaning in reference to an object includes recognizing multiple attributes of an object. Also objects can be identified in spatial relation to other objects in its surrounding. For example, Selvaraju et al. (2020) attempt to enhance a VQA model visual grounding through asking sub-questions about the object's attributes such as color and size and spatial relations. Our hypothesis, in the third research question, is similar to the latter. We will later test if asking these questions would actually improve the visual grounding by means of training the robot to recognize these attributes through asking questions.

4.1.2 How are Questions Generated?

Extending the data-set is in the form of asking more questions about the objects found in EQA-MP3D. We add two different types of questions, spatial and size questions. To achieve this we use existing navigation paths and create new episodes by merging new questions with these paths. As mentioned in previous sections, the episodes are executable functions; when inserted in an environment, they yield an answer. Therefore, our newly generated episodes are constructed in similar structure to EQA-MP3D so that they are all executable within the environments.

The questions' strings are automatically generated in templates. There are two template variants for each general question type (size, spatial)—one variant with a room specified in the question and the other without a specified room. The one with room specified is for generating a question of an EQA-MP3d episode of *color_room* type, such that a question like "what color is the table in the room?" would have a corresponding size question "how big is the table in the living room?". Templates without specified room are for generating new questions of the episodes of *color* type in EQA-MP3D, such as "what color is the table?", and a new question would be "how big is the table?".

Generating Size Questions The templates for size questions are as the following:

- size_obj: 'how big <AUX> the <OBJ> ?
- size_room: 'how big <AUX> the <OBJ> in the <ROOM>?'

For generating a size question we fill the templates automatically with entities, only an object is filled in *size_obj* template and an object and a room in the *size_room* template. The *<AUX>* is an auxiliary verb in the present tense and it's filled automatically depending on the verb-subject agreement; the *<AUX>* takes *is* if the object is singular and *are* if the object is plural. The object's info such as *name*, *ID*, *room ID*, where the object is located, and the geometry of its AABB are extracted from an EQA-MP3D episode.

Size Answer Generation The next step consist of generating an answer to the questions. The answer is determined based on the the object's OOBB size. The size of the object's OOBB is extracted from semantic annotation of an MP3D environment.The answer is generated depending on a criteria that we describe in detail in the following sections.

The final step consist of merging navigational data with a question-answer pair to form a new episode. The object's info such as *name*, *ID*, *room ID* and AABB geometry are also inserted in the new episode.

Generating Spatial Questions The spatial questions are of three relational types. A spatial question can either ask if there is an object *next to*, *on* or *close to* other object. For each relational type there two variants of templates. The templates are the following:

- '<AUX> there <ARTICLE> <OBJ1> close to the <OBJ> in the <ROOM>?'
- '<AUX> there <ARTICLE> <OBJ1> next to the <OBJ> in the <ROOM>?'
- '<AUX> there <ARTICLE> <OBJ1> on the <OBJ> in the <ROOM>?
- '<AUX> there <ARTICLE> <OBJ1> close to the <OBJ>?'
- '<AUX> there <ARTICLE> <OBJ1> next to the <OBJ> ?'
- '<AUX> there <ARTICLE> <OBJ1> on the <OBJ>?'

For spatial questions, the templates are filled with two objects. The *<ARTICLE>* is assigned automatically and it can be either *a* or *an* depending on the agreement with object. *<OBJ>* is an object we find in an EQA-MP3D episode, and *<OBJ1>* is a second object extracted from semantic annotations of an MP3D environment; such as a question "is there a chair next to the table?" where "table" is an object in an EQA-MP3D episode and chair is a new object found spatially related to the table. *<OBJ1>* can be also a random object of no relation to *<OBJ>* which in such case the answer for the question would be "No". when *<OBJ1>* has a positive spatial relation to *<OBJ>* the answer is "yes".

Generating Spatial Answers The data for determining a positive spatial relation between an object and the object we find in an EQA-MP3D episode is extracted from the semantic annotations of the MP3D environments. The criteria for estimating spatial relations is described in further sections.

After generating a question-answer pair we then emerge the pair with the navigation data to form an episode. We insert the info of the object (*name*, *ID*, *room ID*), *<OBJ>*, and AABB size, into the episode. We insert into the episode the same information of *<OBJ1>* if the object has a positive spatial relation, question-answers with "yes" answers. Otherwise, one object with a pair with "No" answer is a random object with no existence in the scene.

4.1.3 Work Structure

The work structure of generating question-answer consist of two components:

- The first component is a **parser** that has two main functionalities:
  - The first functionality is doing data extraction and processing.
  - The second is non-parsing functionality of estimating spatial relations, which is used for generating spatial answers.
- The second component is the **question-answer** episode **generator**.

The next section begins with describing the parser followed by the spatial estimator, then the question-generator, and ends with presenting results. Description of the parser is split into two parts; The first part shows the process of extracting semantic annotations, and the second part views how the spatial relations are estimated.

The section, after that, moves to describe the workflow of the generator and the steps taken for constructing an episode from a generated question-answer pair. Finally, the section ends with results showing the number of questions generated for each question type and the answer distribution in the extended dataset. The results section ends with a discussion around the semantics of the questions asked. The discussion raises questions about the precision of the conveyed meaning in the question-answers and how the meaning might be perceived.

4.2 Data Parser

The data parser is initially used to parse the semantic annotations, process geometric data, and save it to generate answers. The second functionality of the parser is to act as spatial relation estimator simultaneously used while generating questions. These two functionalities are divided into two components. We begin with describing the first component, the annotation extractor, which includes two different experiments/ways of extraction. The description of the second component, the spatial estimator, includes the measurements from which spatial relations were determined for pairs of objects.

The first experiment for extracting semantic annotation is extracting from 'house files' of the MatterPort3D (MP3D) dataset. The second experiment uses Habitat Simulator and sensors. The annotations extracted through Habitat Simulator provide additional computed information. In addition, the objects names in the semantic annotations found in MatterPort3D is different from the annotations given to the object in Habitat Simulator. The names the robot/sensors see in the simulated environment are categorized differently; for example, object names in MP3D such as l-shaped sofa and rounded-sofa are transformed, in Habitat

Simulator, into their Hypernym category 'sofa'. However, the rest of information, such as *object IDs* and location-centers, are consistent with the annotation of the MP3D.

In the existing generated question-answers dataset, we use the data extracted through the Habitat semantic sensors. The main reason for choosing Habitat Simulator's sensors is because they provide computed geometric information of the object's Axis Oriented Bounding Box. The MatterPort3D annotation files include only the radius of objects' OOBBs. An additional important reason for this choice of extraction is the names of the objects extracted by the sensors align with the names found in the original EQA-MP3D task dataset. Choosing object names that align with names found in EQA-MP3D helps have the overall data consistent with each other when we emerge our generated questions with EQA-MP3D.

4.2.1 Direct Annotation Extraction from MatterPort3D Files

As mentioned in earlier section, the semantic annotations of the environments can be found in "X.house" file. The annotations of every object is marked by "O" line. Every line that begin with an O letter consist of one object in the house with a corresponding information about its geometry and location within a room and level-floor. Each "O" line looks as such: [O object_index region_index category_index px py pz a0x a0y a0z a1x a1y a1z r0 r1 r2 0 0 0 0 0 0 0 0 0]

We extract two types of raw information from each object's line of annotation found in the "house" files in MatterPort3D (MP3D). We obtain the object and room IDs, the radius of the of OOBB labeled as *[r0 r1 r2]*, and the center of the AABB/OOBB labeled as *[px py pz]*.

We structure the data in a hierarchical form that the annotation of a house begins with the first level in it, followed by the rooms and objects in each room as house 1 [level1:room1[bedroom]:(obj1:bed,obj2:..),room2:(obj..),, level2:.....]. The extracted data is then saved into a file.

4.2.2 Annotation Extraction Using Habitat Simulator

Our final choice for extracting semantic annotations is Habitat's simulator. Our annotation's parser of the houses uses the sensors with configuration provided by the habitat platform [13]. The configurations include the settings such as the scene, the height and width of the sensors, and the types of sensors to include. The extraction includes using color sensor, semantic sensor, and depth sensor. The sensors are configured at a certain height and width so that they resemble an actual visual source of a physical agent.

Once we simulate the environment, we obtain the annotation as a raw data. We parse the data to obtain information about the levels, rooms, and objects in the rooms. We freeze the simulator after the annotation extraction of of one environment is complete , then repeat the process for the other environments.

In addition to the semantic annotation, we extract the center, radius of each of the AABB and OOBB of the objects. The radius size of the AABB is computed within the simulator and extracted with objects annotations. The radius of the OOBB is used for finding the objects' sizes. The center and radius of the AABB are processed into a piece of information useful for a method of estimating spatial relations among objects.

After the raw data is extracted we process it and **calculate corners of the AABBs** of the objects. The second step is **sorting and saving the data** for question generation. The two steps are described in detail

[13] `https://aihabitat.org/docs/habitat-lab/habitat-sim-demo.html#scene-semantic-annotations`.

in the following:

Calculating the Min and Max Values of AABB Corners The center and radius of the AABB are used to find position points on the edges of the object. Knowing the borders of an object's AABB provides a way to determine a spatial relation between objects given the distances between the corners of two objects. The first information we obtain from the center and radius of the object is two corners of the AABB. Figure 15 illustrates an AABB in 3D as the AABBs we get with the objects annotation. Each of R_x, R_y, R_z is 1d radius on the x,y,z-axis, where the x is the length, y is the width, and z is the height. The radius in 3D would be the line/vector from the center (C) to either *Min* or *Max*. The *Min* can be described as the position point stretched from the center by the length of the radius on the negative direction of all the axis, and the *Max* is the point on the positive direction of the center, at the end of the radius length.

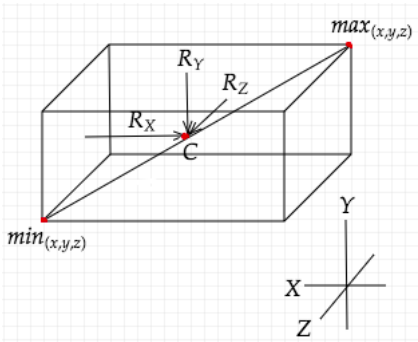


Figure 15: Min and Max of an Axis Orients Bounding Box. R_x is the radius on the x-axis, R_y is the radius on the y-axis, R_z is the radius on the z-axis. The line from C to Max is the radius in 3D, equal to the line from C to Min. The line from Min to Max is the 3D diameter. Radius is half of the diameter.

The first calculation is finding the *Min* and *Max* points of a bounding box given an object's center and its 3D radius.

The AABB radius extracted from the habitat simulator is in diameter form as the line from Min to Max. The radius would be half the extent of the diameter, so we get the 3D radius by simply dividing the diameter by two. $Radius = D_(x, y, z)/2$. Next we calculate the *Min* and *Max* points of a bounding box given an object's center and its 3D radius.

Since the center is a point at one end of the radius, and the radius is a vector, calculating the *Min* and *Max* is done by subtracting or adding the 3D radius from the 3D center; if we add the length of the vector to the center point, we get the point at the end of its length *Max*. If we subtract the radius length from the center point we get the point at the end of the radius length in the minus direction which is the *Min*. Below, C denotes the center point, and $\vec{R}$ denotes the radius as a vector.

$$Min\ point\ =\ C\ -\ \vec{R}\ =\ (x_1 - x_2, y_1 - y_2, z_1 - z_2)$$

$$Max\ point\ =\ C\ +\ \vec{R}\ =\ (x_1 + x_2, y_1 + y_2, z_1 + z_2)$$

The *Min* and *Max* as corners of the box could be used to estimate distances between objects. For example, in 3D game design, they are often used for collision detection (Cai et al., 2014). From the Min and Max, one could obtain the values of all the other corners, as the values of the other corners are a distribution of the [Min, Max] of the object's points in all axis. In the following sections, we describe how Min and Max are used in a technique for finding spatial relations between objects.

For every object in the annotation, we find the Min and Max of its AABB and extract the radius of its OOBB. The radius of OOBB is given with the data extracted in the simulator. We use the OOBB radius for calculating the sizes of the objects. We consider the size as the box's volume, which is the length multiplied by the height and width. In our case, the length is the *x* delimiter, and width is the *y* delimiter, and height is *z* delimiter. The calculated volume of a box is $X \times Y \times Z$.

Sorting and Saving the Annotations We sort the annotations and save them in a file. The data is structured hierarchically. At the top part is the *house ID*, then rooms in the house, then the objects in the house. Each object is sorted by id contains the *Min* and *Max* value of its AABB, *radius* of the OOBB, its *name & ID*, *room name*, and the *level Id* where the room is located. Structuring and processing the data and storing it in files allows access to all the objects in a room through indexing the scene id and room id, which accelerates the question generating process.

We store the calculated volume of each object in all the houses in a second file. The volumes are stored in a dictionary of object's categories. For example, the volumes of all the sofas in all the enviroments could found in the category 'sofa'. The point here is to obtain data on the overall sizes of each object type. We use this information for finding ground-truth answers for size questions. Generating answers for size questions is elaborated in detail in further sections.

4.2.3 Spatial Relation Estimator

The estimation of spatial relations is done by taking a group of objects and pair them according to spatial relations. All the objects in a room are passed to a *spatial estimator* to find pairs that are 'on', 'next to' or 'close' to each other. If the mentioned relations are identified between two objects, the pairs are sorted by type, each type is a spatial relation. This information is used for generating positive spatial questions.

The first measure for determining the mentioned spatial relations is by calculating the distance between the corners of AABBs of two objects. In the processed annotations, the objects are initially represented by two corners, "Min" and "Max", as seen in a previous section. The other corner points of the box can be extrapolated from the "Min" and "Max", as certain dimensions overlap.

The spatial relation estimator from the Min and Max points replicates parts of a source code published with the EQA paper (Das et al., 2018) [14]. The spatial relation estimator is part of a question-answer generator that was constructed for generating QA for different 3D environments than MP3D. The code is available on the EQA official website, and we use the same methodology of estimating spatial relations between objects [15]

In the following text in this section, we begin with describing two operations for measuring distances between a pair of objects which are **extrapolating AABB corners from Min & Max points** and **measuring distance between corners of two AABB's**. The third operation is **classifying relations between the sidelines of AABBs** which is generally used in defining specific relations. The text then describes the specific conditions and criteria used for defining each of the *on*, *next to* and *close to* relation.

[14] Link to the code source used for estimating spatial relations `https://github.com/facebookresearch/EmbodiedQA/blob/main/data/question-gen/house_parse.py`.

[15] Question generation code for an erlier version of EQA can be found at EQA official website`https://embodiedqa.org/`.

Extrapolating AABB Corners from Min & Max Points In Figure 16 we see and illustrations of the eight corners of an AABB. If we move our point of view directly in front of the cube as if we are facing the square *GHED*, the points A and H would seem to be lying in a straight line. Lying on the same straight line, for example, means the point A and H are located on the same points in the x-axis, and so one for the other parallel points.

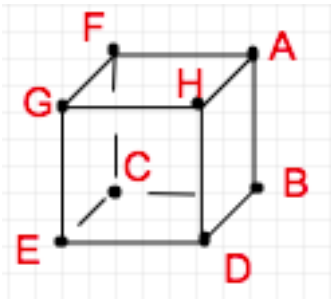


Figure 16: Corners of Axis-Aligned Bounding Box. The box viewed here from upward rotated to the right position from the front of the box. The correct global viewpoint of the box would be by imagining our viewpoint straight facing the square GHED, where A&H would be on a straight line(similar points on the x-axis), and as such for all other parallel points. From the illustrated corners, *A* would be the "Max" and *E* would be the "Min".

We get the rest of the points from the Min and Max of an AABB because AABB's are not rotated and aligning with the global view. To express it better, we imagine the global point of view of the AABBS as a view facing a group of ordered objects facing the same reference line, not rotated. When the axes are aligned, the values of the corners overlap where the 3D values on the (x,y,z) would be either the 'Max' or 'Min' in each dimension. For our example in Figure 16, the point *A* represents the "Max" corner point, and the point *E* represents the "Min" corner. We can extrapolate, from the "Min" & "Max", the six other corners as such: $A = (x_{max}, Y_{max,} Z_{max}), F = (x_{min}, Y_{max,} Z_{max}), H = (x_{max}, Y_{min,} Z_{max}),$ $B = (x_{max}, Y_{max,} Z_{min}), D = (x_{max}, Y_{min,} Z_{min}),\ C = (x_{min}, Y_{max,} Z_{min}),$ $G = (x_{min}, Y_{min,} Z_{max}),\ E = (x_{min}, Y_{min,} Z_{min})$

Measuring Distance Between Corners of Two AABB's The first criteria for determining a potential pair with spatial relation is the distance between their corners. The Euclidean distance between two corner points; denoted as the distance between p and q, where P is one corner of an object and q is the corner of the second object. *n* denotes an Euclidean space, q_i&p_i are the Euclidean vectors of the corners where the denominator *i* stand for the dimensions of the vector. The formula can be described as the square root of the sum of the square of the subtractions of q and p at every *i*-dimension.

$$d(p,q) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

Depending on the type of spatial relation we want to detect, the corners can be represented as points in 1d, 2d, or 3d. For example, to filter pair of objects where one is *on* the other, we would check how close they are on the 3D axis, but then we want to know the distance on the z-axis(the height) in particular. Therefore, the calculated euclidean distance between the corners in 1D as such: $\sqrt{(z_1 - z_2)^2}$. Knowing the distance on the x and y-axis would be indicative of corners next to each other; in such a case, the measure of the distance of 2d corners would be as such:$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$. We would represent the corners in

3D if we wanted to measure how close two corners are to each other in general, not on a specified axis. The Euclidean distance between two 3D points would be as such: $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 \ + \ (z_1 - z_2)^2}$

Classifying Relations Between the Sidelines of AABBs If a pair of objects are in close distance, we distinguish the possible spatial relations they have depending on the overlap of the sides of their AABBs. For example, in a case of nearby boxes that might be *on* each other, the vertical line ($Z_{min} - Z_{max}$) of one object's box should not be contained within the vertical line of the other.

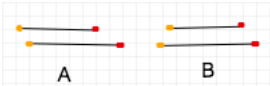


Figure 17: (A) & (B) are examples of boxes' sides on the x-axis. In (A) the top line is not contained within the lower line. In (B) the top line is contained. In each line, the yellow point is the x_{Min} and the red point is the x_{Max}.

The calculation if one side is contained within the other relies on defined criteria. In the drawing 17 *A* represents two lines on the x-axis where the top part is not contained within the other, and in *B* the top is contained within the lower line. In this example, the *contain* relation is determined by taking the 'Min' represented by the orange dot and the 'Max' doted in red. Blow, *U* denotes the upper line and *L* denotes the lower line. The upper line is considered contained within the lower in **B** given the following function:

$$Contained\,(U, L) \quad Iff \quad Min_U \ > \ Min_L \ , \ Max_U \ < \ Max_L$$

In words, the operation above states that the upper line is contained within the lower line given the following conditions:

1. If the Min of the upper line is greater than the Min of the lower line.
2. If the Max of the upper line is less than the Max of the lower line.

The *contain* operation above is done over different axis for the *on* and *next to* relations. Below we specify how each of the *on*, *next to*, and *close to* relation is determined between the objects, after a pair of objects are selected given a distance.

Defining the *On* Relation

- The first step is choosing pairs of objects closest to each other vertically (on the Z-axis). The two boxes should be touching on the Z-axis. The Euclidean distance is calculated between the 1D points on the Z-axis only, and the two touching objects should have a vertical distance within 0.5 millimeters. We believe a distance with 0.5 would perceptually appear as objects touching each other.

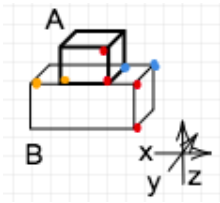


Figure 18: The vertical line between the red dots is the height and can be defined as being between z_{Min}&z_{Max} of a box. The lines between the yellow and the red dot is the width and is defined as being between x_{Min} and x_{Max}.

- The second step consists of a group of conditions that the pair of objects need to meet to be considered *on* each other.The conditions are as the following:
  1. The first condition is that the vertical sides, the line from Z_{min} to Z_{max}, of the boxes are not contained within each other. The $Z_{min} - Z_{max}$ lines of every object box are the lines between the two red dots in box *A* and *B* in the illustration 18. Otherwise, if the lines on the Z-axis are contained, it would mean one object is inside the other.
  2. The second condition is that the horizontal line of one of the boxes is contained within each other.
- The final step is deciding which object is on top of the other. The pair of objects are passed to a function that determines which $Z_{min} - Z_{max}$ has a greater value. The object on the top should be in the positive upward direction.

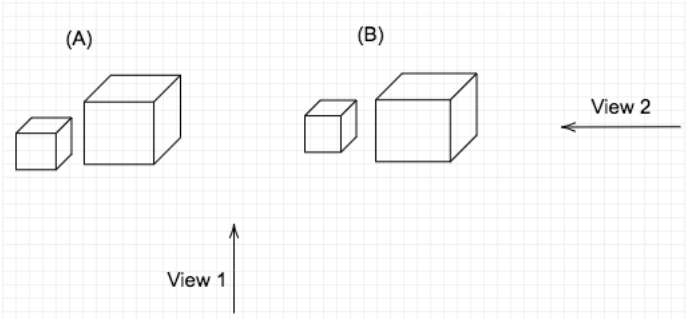


Figure 19: From view 1 the pairs A & B can be both considered as next to each other. From view 2 the pair B seem as one box is behind the other not next to it.

Defining the *Next to* relation

- the first condition is that a pair of objects next to each other have to have a distance not greater than 0.1 meters on the X&Y-axis. The distance here is calculated for 2D corner points; this means the distance is calculated for four corners (Min and Max front and back). The reason for deciding this distance is because if the objects are not close enough to each other we might not be considering them next to each other. A second reason for not enlarging the distance is to ensure that two objects are seen next to each within the sight of the robot in the scene.
- The second condition, the pair should have contained sides in neither the x nor the y axis. In this condition, a pair of objects next to each other would look like illustration *A* in Figure 19. This *next to* relation might a bit different from what we consider next to each other as humans. We might imagine a typical next to pair as illustration B seen from view 1.

However, the choice of having 'next to' pairs not contained with each other is due to considerations of the viewpoint. From view *1*, the pair(B) seem next to each other, but from view two, they would not. Pair (B) from view *2*, one object would be behind the other and likely hidden. assigning *next to* pairs as in illustration *A*, the pair would be still visible in whichever view.

- In the final condition, the pair must have their lines overlapping on the z-axis. Otherwise, the two objects might satisfy the first condition on the (x,y) but be distant on the z-axis, such as one object in the ceiling and the other on the floor.

Close to A pair close to each other are a pair that has any of their 3D corners close to each other within a distance between 0.2 meters and 0.25. Limiting the second object's distance to 0.25 meters is to ensure that the object is within the sight of the robot.

4.3 Question Generation

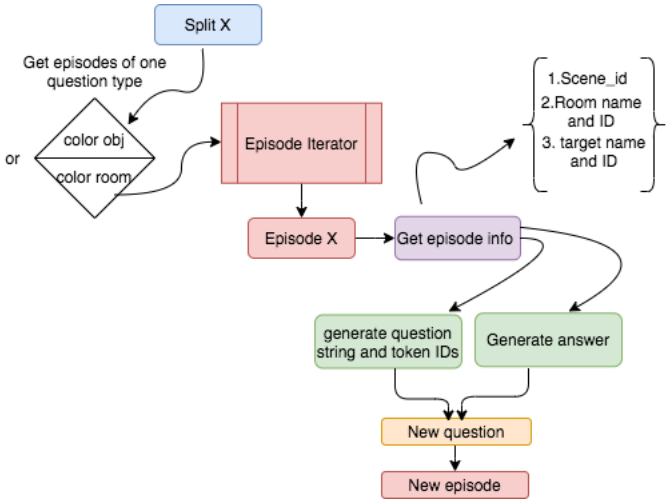


Figure 20: Split generator: "Split x" can be either train or validation set of EQA-MP3D. The rotated uncolored square is a filter that picks either episodes of color or color_room. Red rectangle with two lines iterates through the filtered set. The purple rectangle is a parser that extracts object's info and navigational data seen in the curly brackets pointed by an arrow. The green boxes generate a pair of QA. The bottom yellow and red rectangles mark the process of transforming a pair of QA into an episode with navigation data.

Our generation of QA episodes replicates the episode format of EQA-MP3D. The question-answer generator turns an EQA-MP3D split of episodes into a new split of new QA episodes. In Figure 20 we see an illustration of the workflow of the episode generator. The five general steps, as seen in Figure 20, are as the following:

1. Filtering(Uncolored rotated square).
2. Iterating Episodes(Red rectangle).
3. Parsing Episodes (Pink rectangle).
4. Question-Answer Generation (Green rectangles).
5. Episode Wrapping (The bottom yellow rectangle- Inputs QA and outputs episode).

The first step, filtering an EQA-MP3D data-split, picks episodes of one type-variant. For example, to generate question of "size_room" we filter EQA-MP3D and take the episodes of "color_room" only. Second, each episode in the filtered set is iterated and passed to the parser. The parser copies the navigational data and the info about object in question, where the navigational data is copied into the new episode and object's info is used to allocate the object's metadata in the annotations. The object's info taken from an EQA-MP3D, in particular, is the object name and id, scene ID and room ID. The object's metadata in the annotations provides the information needed for generating a new question-answer pair.

Generating a pair of question-answer consist of producing a question token&ID and an answer. Each question-token is generated in the template assigned for the specific type&variant. The question answers, on the other hand, follow different conventions depending on the question type. Generating an answer for spatial questions would, for example, require finding an object with spatial relation to the target object found in an EQA-MP3D episode; and generating a size answer relies on calculating the objects size and compare it to the size of the other objects of its type.

The final step consists of inserting the new question with the corresponding geometric information, into an EQA-MP3D episode format. We call a QA sample an episode when the section of the episode seen in Figure 14 are filled with the new QA and the other the corresponding information.

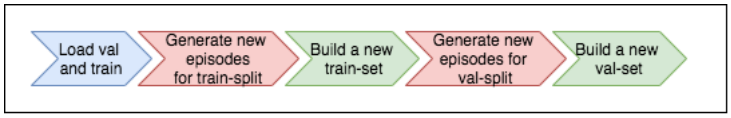


Figure 21: Episodes for train are generated first, then building the set is complete by inserting with answer and question vocab. The validation split is generated and loaded next

The train&validation splits are generated and loaded subsequently. As seen in Figure 21, the first split generated is the train split then the validation split. The loading/building of the split is inserting the top most layer of the set which is the answer and question vocab. The reason for generating the two splits in two different stages is for not mixing the environments and scenes between the validation and the training sets. Generating questions for the train and validation sets separately ensures that the robot would be trained and tested on different environments.

Generating the two splits separately also helps to keep track of the answers distribution for each split. The current code controls the distribution of answers in each of the train and the validation sets.

4.3.1 Size Questions

Size questions have three possible answer choices, *big*, *small*, and *medium*. Each answer is determined by the volume of the object's OOBB relative to the volumes of the other objects of its category type in the environments. Volume of a the OOBB is the width times the length, times the height: $OOBB\ Volume = W \times L \times H$. The first step in generating a size-answer is calculating the volume of the object's OOBB, and the second step is to compare it to a standard size of its category.

In the second step, the relative size is determined by a volume's deviation from the standard size of its type. The object is considered an element of a size set if its within the range assigned for the set. The range and borderlines of each set are determined in relation to a context. Establishing a range for a vague expression relative to a context is defined by Raffman (1996) as multi-range theory

"For any object O, vague predicate 'P', and total context TC: 'P' applies to O, relative to TC, just in case a

competent speaker could judge O in TC and, were he to judge it in TC, he would apply 'P' to it."(Raffman, 1996)(pp.181)

The context in this project has been defined by the category of objects found in all the 3D environment; For example, the context of "big sofa" is all the sofas we find in MP3D. The standard deviation can be seen as the established boundaries taken from the contexts. The range of deviation of each object category is different and respective to its context, so that the size range of "big sofas" is different from the range of "big fireplace".

As mentioned earlier the sizes of all objects are sorted by category in a file. We access the volumes of an object's category and calculate the mean size and the standard deviation of the sizes of an object's category. The formula below denotes the calculation of the standard deviation and it includes the following denotations: μ denotes the *population mean* which is the sum of the volumes divided by all the number of items. Second is the *variance* denoted as such:

$$\frac{\sum(x_i - \mu)^2}{N}$$

Variance is the average of the squared differences from the mean where x_i is the $i - th$ item and N is the total number of items. The standard deviation is the square root of the variant.

$$s = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

To calculate the standard deviation of the volumes we first get the mean then the variance then the square root of the variance. For example, if we have three items (4,7,2) we calculate the mean as such $\mu = \frac{(4+7+2)}{3} = 4.3$, then the variance: $V = \frac{(4-4.3)^2+(7-4.3)^2+(2-4.3)^2}{3}$, and the Standard Deviation would be the square root of the variance, $S = \sqrt{V}$.

The answer is "small" if the object's size is smaller the mean size of its type minus the standard deviation, "big" if the size is larger the mean + the standard deviation, and "middle" if the size of the object is within the standard deviation added and subtracted from the mean. This method of judging sizes might not qualify as a competent speaker. However, it approximates the possible judgments of a competent speaker. The latter is based on the assumption that the speaker's experience is confined within the world of the environments. The speaker judgments of sizes, would, thus, be dependant on the varying sizes of the object categories that the speaker is exposed to within the existent environments.

We control the answers' distribution. We observe that a majority of objects have a mean size given the standard of their type. In order to avoid bias towards the 'medium' answer, we restrict the number of QA with medium answer. We keep track of how many QA with medium answers has been generated and when the number reaches a limit we generate None QA that are later filtered out. The limit varies depending on the question variant (Questions with or without "room" string), and the split (train or validation). The limit values are assigned based on observations of the answer distribution in the splits.

4.3.2 Spatial Questions

Generating spatial question takes more complex steps and longer time than generating size questions. Generating a spatial QA requires a coordination with the spatial relation extractor. In addition, spatial questions include the addition of an extra object to the question string, and the insertion of the new object's information into the QA episode.

Searching for spatial relations of the target object in an EQA episode is the first step taken. We pass the scene and *room ID* to the relation extractor to obtain pairs of objects, within a room, with a spatial relation between them. The relation extractor could define three types of relations : *next*, *on*, or *close*, if existent within a room.

The decision of generating a question of one of the mentioned relational categories is dependent on the existent of an object with a spatial relation to the target object. The process of executing a generation command of a question of a spatial type is illustrated in Figure 22. If there is an object ”on” the target or a target is on another object, we generate one questions, and similar case if there is an object next to the target object. If there is no ”on” or ”next” relation or either of them is non existent, the criteria for checking if there is a ”close” object is satisfied. If none of the conditions are satisfied a QA with ”no” answer of a random spatial type is generated.

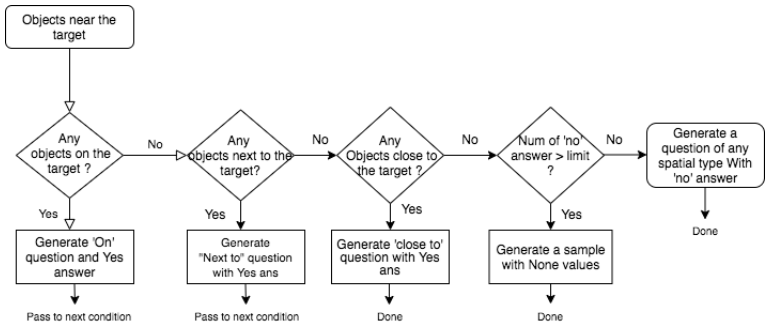


Figure 22: Decision tree for generating different types of spatial questions.

A QA with positive spatial answer has a ”yes” answer, and ”no” if a relation is non existent. The decision tree as seen 22 leverages positive QA for the reason that we observe that the no-relation instances outnumber the positive ones. The final condition, we even control the number of QA with ”no” answer by generating a None QA if the number of generated QA with no answer reaches a limit. The QAs' with None values are later filtered out.

Within this decision structure, for each *navigational data* in an EQA episode (for each episode), there is a possibility for generating from one to two spatial questions of different spatial type.

The process of generating a spatial question includes the addition of information about two objects. An episode/question generator, a group of functions, adjust itself to a spatial question generation if certain arguments are given to it.

4.4 Results

4.4.1 Total Number of Generated Questions

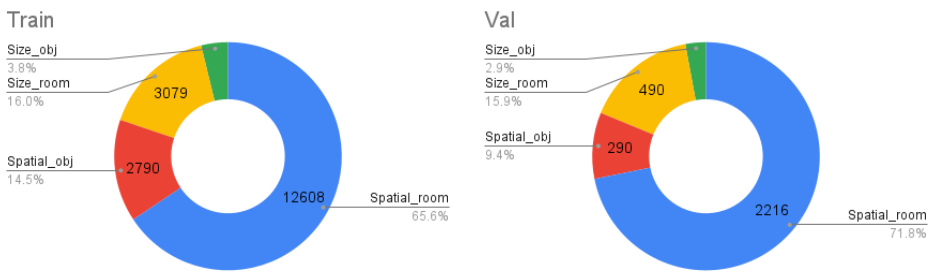


Figure 23

We generate a total of 19 207 question for train and 3 186 questions for validation. In Figure 23 questions of size_room and spatial_room refer to questions that contains a reference to a room, such as 'How big is the bed in the bedroom?'. Questions of spatial_obj or size_obj type are questions with a reference to object only, such as "Is there a chair next to the table?".

4.4.2 Answers Distribution

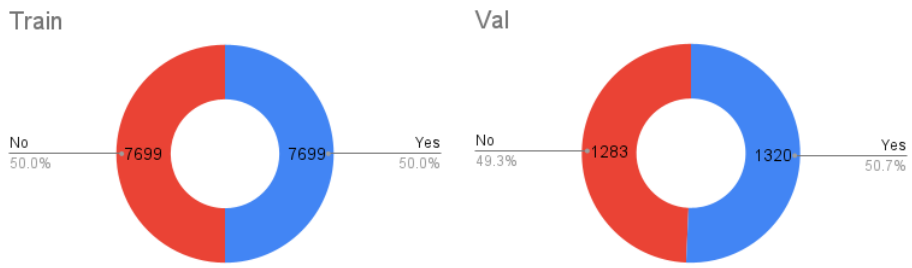


Figure 24: Number of question generated per type.

Answers Distribution of Spatial Questions The generated spatial question has equal answer distribution as seen in Figure 24. The balancing of this answer distribution has been controlled manually by observing the number of positive relations extracted, and limiting the number of negative instances to equal the number of the positive ones.

The chances that a spatial relation between two objects is negated is achieved by generating "No" QA using the same objects asked about in "Yes" QA. The latter means every two objects with positive spatial relations have a high probability of being asked about in a negative spatial relation situation. In this regard, a balanced data set would contribute to positive learning outcomes.

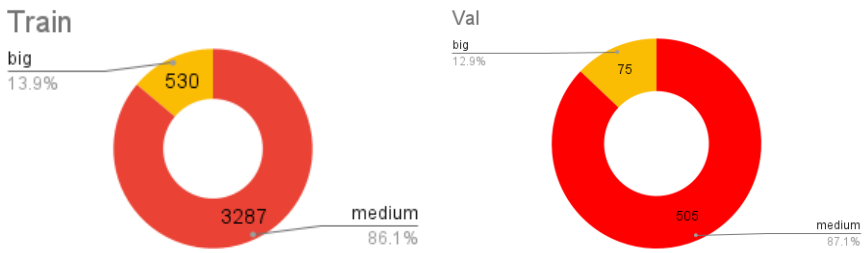


Figure 25

Answers Distribution of Size Questions The generation resulted in zero samples of "small" answers and a majority of 'medium' answers as seen in Figure 25. In the QA generator, we intended to limit the question-answers with "Medium" answers based on an observation of their dominance. However, limiting the 'medium' answers more than the presented numbers would have resulted in very few question-answers of a size type. An insignificant proportion of size questions was insufficient for training the model. We decided to keep the size questions with their imbalanced distribution, despite knowing that this linguistic bias might hinder the learning outcomes.

4.4.3 Discussion

Choice of Annotation The main reason for choosing the annotation extracted from Habitat-simulator is to have generalized textual references for the same visual data. As mentioned earlier, the annotation extracted directly from MP3D has more hyponyme categorizations of objects instead. Introducing these new names of sofa to the model in the newly generated question might confuse the model. The confusion in learning might happen as all these types of sofas are referred to in ”color” questions as only ”sofa” wherein the new question ”instances” would be referred to differently. Unifying all the object names would tell the model that we are referring to the same object in the different question types.

One could argue that it would also be good if the model learns that ”l-shaped sofa” also means ”sofa”. However, the focus of this work is not to teach the robot the relation between hypernemes and hyponymes nor introduce new objects to the model. The main focus of this work is to improve the model's attention to vision and examine its ability to answer color questions after training with new questions. Having similar objects in all question types help in narrowing the focus of the training. When the objects are named similarly across different question types, it would contribute to the model's establishing a link between the linguistic reference and its visual representation. Shah et al. (2019); Ray et al. (2019) are two examples of modified VQA datasets that contain the addition of more questions that ask about the same object using the same name categorizations. The researchers note that consistency in asking different questions about the same entity contributes to the system's visual grounding.

Another benefit from using the same object-naming is to avoid data biases in the newly generated spatial questions. The spatial QA with a ”No” answer selects a random object from a set of previously selected objects for a QA with positive spatial relation. When the set of objects to choose from is restricted(object names in hyperneme means fewer object names in the set), the likelihood that an object-name in a ”Yes” QA be selected for a ”No” QA is higher. Kafle et al. (2017) use templates and similar object selection methods as ours to generate new questions to balance a VQA dataset; their approach to achieving a balance relied on using the same annotations/object-names. The mentioned source notes a similar observation as ours. Using the same names in new questions decreases the bias because if we use the same QA for a different image, a question string would be negated in a different image.

Vagueness & Ambiguity of Textual References in Relation to Perception The validity of our method of connecting the meaning ”big sofa”(intention) to the extension it refers to is vague in two regards. *vagueness* is the higher-order vagueness described by the vagueness appearing in one percefication above or below the drawn borderline. For example, two big objects appear similar in size. One would be classified as medium size for being within the standard deviation, and the other would be classified as *big* for being only one size unit above the deviation.

The second existing vagueness is the description of sizes relative to the robot's distance from the object. Does the description of 'big chair' match what we believe is a 'big chair' seen from the robot's distance to the object (the extension)?. No, we do not know if each size category appears in a similar approximation relative to similar viewpoints from the distances that the robot stops at.

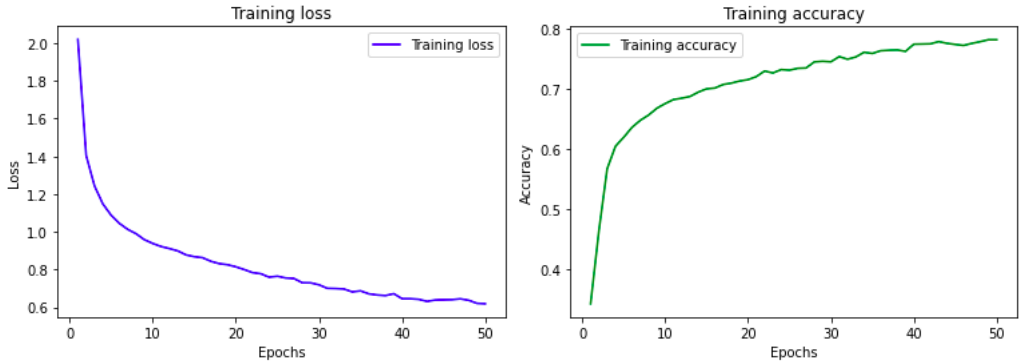
The measure for expressing spatial relations reduces vagueness. Generating spatial expressions undertook stricter defining measures than sizes. The measure taken included consideration of viewpoint and more precise rules for determining the membership of a set of spatial relations. For example, for two objects to be included in the set of objects ”next to” each other, the two objects have to fulfill extended geometric criteria.

However, Spatial relations could be ambiguous. A pair of objects considered ”on” or ”next to” in a few

instances can also be considered close to each other. However, this should be no issue for hindering learning. In some cases, the robot could learn that next to, in some cases, can also mean ”close to”. It might be an issue if our QA asked for specifying the relation, such as asking what relation two objects have, where the model would answer that they are close to each other. However, we expect a more specific answer such as ”on” or ”next to” each other. Since our spatial questions are yes/no questions, this leads to no ambiguity concerning the answer.

5 Task Two - Question Asking and Answering

5.1 Training



(a) Loss (b) Accuracy

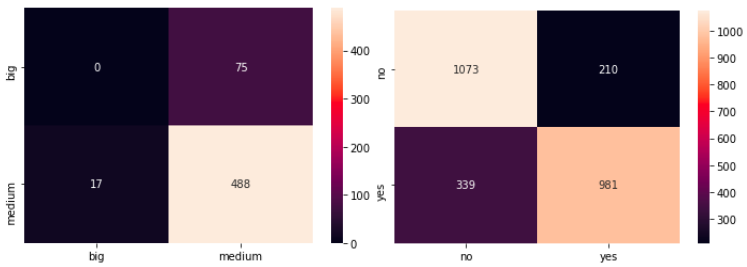
Figure 26: The dropping loss on the left indicates learning progress, as prediction accuracy consequently increases as seen to the right.

We train the same CNN-LSTM VQA model on the new and old questions with 50 epochs. In particular, we train the LSTM with attention to the visual features and use a pre-trained CNN for encoding visual features. The pre-trained CNN we use is proposed by the researchers in the Habitat platform and could be found on the EQA Github page.[16].

The learning of the model throughout the epochs is improving. The is indicated by the decreasing loss in every epoch as seen in Figure 26. For evaluation, we pick the model trained with 50 epochs. The model at epoch 50 is the best performing model in a range of epochs. As seen in Figure 26, the model at epoch 50 has the highest average accuracy and lowest loss.

5.2 Evaluation

The evaluation of the validation set with all the question types shows an average accuracy of 63%.



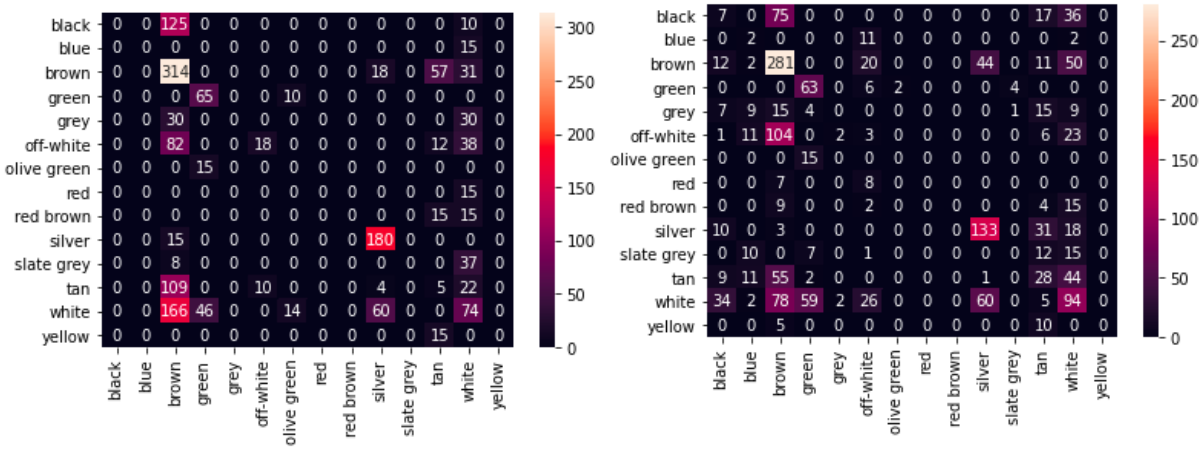
(a) Size questions (b) Spatial questions

Figure 27: (a) is confusion matrix of the predictions of size questions, with an evaluation yielding an 84% average accuracy. (b) is the predictions for spatial questions with a 79% average accuracy in the evaluation.

[16]Link to the code for running the VQA baseline model. The same page include an attachment to the pre-trained-CNN "`https://github.com/facebookresearch/habitat-lab/tree/master/habitat_baselines/il#eqa-cnn-pretrain-model`.

The results of the size questions showed all the predictions to be of 'medium' answer. In Figure 27 (a), the illustration of the predictions shows that all the answers to size questions have been predicted as "medium". The evaluation results of the size questions are not surprising given the significant imbalance of the answer distribution in the training set.

Below we see the distributional difference of color predictions between the original model and the model trained on the new questions. Figure 28 (a) represents a heat-map of the confusion matrix of the model's predictions before training with a new question. The heat-map (b) in Figure 28 represents the predictions after training with new questions.



(a) Before (b) After

Figure 28: (a) represents the evaluation results before training with new questions with 39% average accuracy. (b) is the results after training with new questions with 36% average accuracy. The classes in the columns the are ground truth answers, and row on the bottom are the predictions.

In Figure 28 (a), the squares with the brightest colors show a high consistency of predictions of a certain answer-class. For example, **white** in the column in Figure 28 (b) is a ground truth answer to questions where it's been predicted as **brown** 166 times. An interpretation is that some typical questions in the train-set are biased to "brown" answers. However, in the validation set, the same questions have ground truth "white" answers. Hence- the images are different; the robot is trained and tested on different scenes.

In other cases, the ground truth of typical questions matches between the validation and the train sets, such as the **brown** in the column of (a). Matching question-answer strings would lead to more positive scores in the evaluation. Let's refer to the behavior of concentrated predictions of one class as *prediction consistency* illustrated in the bright squares in Figure 28 (a). This consistency occurs due to bias, and it can impact the scores either positively or negatively.

Training and asking the model different types of questions show a change in the model's behavior in terms of prediction consistency. In Figure (b), the concentration of predictions is slightly more distributed for each color class. For example, predictions of questions with **white** answer, as seen in the column in (b), had more distribution than we see in (a). The increase in answer variations can be seen in almost all the categories in the answer column in (b). The increase in answer distribution after training with the new questions also resulted in lower average accuracy than before training with the new question. The average accuracy of predicting color questions before training on the new questions is 39% , and after training with new questions is 36%.

While predicting more correct answers might superficially seem like better performance, it instead indicates exploiting biases. Less correct predictions but more distributed predictions could mean that the system makes predictions from multiple possible colors in the image— rather than merely leaning on the text priors. Relying on text priors would have resulted in consistent predictions.

However, this prediction distribution could also be occurring randomly. The change of behavior in terms of predicting colors more distributively does not necessarily indicate more attention to the image. The case can be that model learn to give different random answers instead of one answer without considering visual information. Therefore, we conduct two experiments and report their results to measure the model's reliance on vision after training with new questions.

5.2.1 Experiment

We conducted two experiments to investigate whether the model trained with new questions relies more on visual information when answering color questions. The first experiment tests the model blindfolded—the second experiment tests the model with noise instead of actual image features.

The baseline for evaluating the model in the experiments is by comparing the performance before and after manipulating the images. To evaluate whether the model reliance on vision has improved, we experiment once on the original model (untrained with new questions) and the modified model (trained with new questions). The model with more performance changes after the image manipulation is one with more attention to vision. The change in performance could be considered a change of behavior responsive to alteration of the visual information. If changing the visual information consequently affects the prediction, this would indicate more reliance on vision.

In the blindfolding experiment, we give the model non-valuable input consisting of zero numbers. The accuracy in the original model after blindfolding dropped two percent from 38% to 36%. The modified model, trained with new questions, had 8% percent drop in accuracy, from 36% to 28%.

The image manipulation relies on replacing the encoded features of the images with noise. The noisy is randomly selected features from a pool of features from the actual images. The new image features resemble actual images except that they would be unrepresentative of the correct visual features required to answer the question.

The results show that the experiment on the modified model has a greater performance difference than the difference marked in the original model. The original model had an average accuracy of 39% for all the questions. When replaced with noise in the experiment, the results remained the same with an accuracy of 39%. When tested with noisy images, the model trained with new questions resulted in 34% accuracy, dropping 2%.

The drop in accuracy when the modified model, trained with a new question, is blindfolded indicates more reliance on vision. The performance of the initial model has not decreased to the same extent as the modified model. It is reasonable to conclude that training the model on various questions does increase its reliance on vision to a certain extent.

However, giving the model noise instead of an image was not marked by a significant drop of scores as blindfolding it. The latter can be interpreted as the model not relying on vision, unlike the conclusion we make of results in the blindfolding experiment. It instead shows that the model needs some visual stimulus to make more correct predictions.

6 Conclusion & Discussion

Generating questions for embodied question answering is a very challenging task. It requires precision in handling the annotations and consideration that the language generated corresponds with what the robot sees in the environment. The latter includes analyzing our view of the entities in the environments and how the robot might end up seeing it when reaching its navigational destination.

In this project, we defined different views and investigated techniques in which our generated question could be visually and semantically consistent with the visual input that the agent perceives. Our choice of generating size and spatial questions is proposed for the importance of these question types to the robot's practicality and comprehension of meaning. We generated questions using extra-geometric measures. For size descriptions, we used a theoretically supported framework of definition. The framework relies on expressing sizes in gradable measure relative to a context; we defined object sizes in three categories *small*, *medium*, and *big*. For spatial questions, we selected a technique of defining spatial relations that considers the multiple possible views of the scene where the robot would view the target objects. We generated three types of spatial questions *on*, *next to*, and *close to*, with binary answers of "Yes" and "No".

Given the criteria we used for generating size and spatial questions, the generation yielded positive results for spatial questions and negative results for size questions. The generated size question-answeres contain an imbalance answer distribution. For spatial questions, the generation included controlling the distribution of answers, and we achieved a balanced answer distribution.

We trained and evaluated the existent VQA LSTM-CNN model on the new questions. We view the evaluation for each question type and analyse the results. We conclude that the answer predictions for spatial questions had good scores, indicating good learning outcomes. The answer predictions of size questions were completely biased, which is an expected outcome due to the imbalance of the answers in training. Finally, we evaluate and analyse the predictions of color questions and note that the model behaves differently regarding color questions after training with new questions compared to the initial trained model.

Through two experiments, we evaluate the model's reliance on vision after training with new questions. Our hypothesis is that training the robot on size and spatial questions would improve visual grounding. We conclude that the model trained with more questions considers the visual features in its answer prediction greater than untrained with new questions. The latter implies that asking more variant questions to the model does improve its reliance on vision to a restricted extent. We observe that the model gives better predictions if there is a visual stimulus, even if the given visual features are uninformative.

6.1 Limitations

The limited number of visual scenes restricts the number and variety of the questions that could be generated. The number of unique visual scenes for VQA training depends on the amount of navigational data, the paths found in EQA-MP3D, which consist of 658 paths for training. Consequently, having more navigational paths means more visual scenes for VQA training, which would provide richer and a variety of data and objects to ask questions about. In particular, more scenes would make it possible to negate the question of specific spatial relations. In the current spatial questions, a spatial relation between two objects is mostly negated in a different form of relation. To negate the specific relation such as, a book on the table, this would require asking the question in a scene where there is a book on the table and asking the same question in a different scene where there is no book on the table. In addition, having more objects of the same category gives the advantage of having varied sizes of the same object category, thus increasing the richness and balance of the data.

Technique-wise, the most apparent limitation of the work is the method used for generating size questions.

Defining size expression with standard deviation measure results in most objects being within medium range.

Another major limitation stems from the usage of synthetic and extra-geometric measures for defining spatial relations. Defining spatial relations using geometric calculations from the annotations make it possible to define some spatial relations for objects in approximation to what the agent would see. However, the synthetic relation extraction would not be as precise as if the questions were asked with a shared view and distance that the speaker and the hearer have to the objects. In addition, advanced spatial reasoning training might require training on more complex relations with more complex answers.

The existent neural model limitations posses another challenge for training. The variety of questions and their types requires strong attention and mapping between the linguistic references and the visual representations of entities.

Viewing the whole EQA task from a pragmatic position, the agent's capacity to exhibit intelligent linguistic behavior remains confined because it trains and communicates only a few shots. A more linguistic capability can be exhibited in the agent's ability to hold a dialogue.

6.2 Future work

- A considerable improvement for the current generated questions is researching different methods for identifying size expressions.
- More scenes and navigational paths would be very resourceful for generating more and diverse questions. However, such an objective would perhaps require co-corporation with the research field in robotics and navigational training.
- Adding attention or replacing the current CNN-LSTM Visual-Question-Answering model with an attention-based one would hypothetically enhance the system's Visio-linguistic capabilities

7 Ethical Considerations

Training embodied agents in simulated environments causes less harm to the environment and humans. During the training, particularly in navigation, the agent makes many failed attempts. These failed attempts could cause harm to human workers and the environment. The harm could occur by damaging resources and colliding with humans. On the other hand, training the agents in simulated environments avoids bringing the potential damage caused during the training.

However, simulating environments and training the agent on GPUs could have a high CO2 emission. Strubell et al. (2019) notes that the CO2 emissions produced by NLP models trained on GPUs could have a CO2 print more than a single annual use of a car. The variables for deciding the amount of emission released consist of the source of electricity, the hours, and the computational power required to train a model. In the list of categories in Strubell et al. (2019), NLP pipelines such as parsing would have fewer emissions than neural networks training.

The computing time in this project mainly consisted of data processing and parsing. However, the time for training the model and simulating the environments consisted of approximately five training sessions. Each session took approximately ten hours of simulation and training. The sum of hours using a GPU would be around fifty hours.

The training took more than one session because of errors and mistakes found in the data-set. In future work, the computational load could be reduced by reducing the number of training sessions. The number

of training sessions can be reduced by ensuring that a data-set is finalized for a single training session. The latter would imply more NLP pipeline work but less GPU training, which would be less environmentally harmful.

More importantly, training models using a renewable energy source would release the least co2 emissions.

References

Agrawal, A., Batra, D., & Parikh, D. (2016). Analyzing the behavior of visual question answering models. *arXiv preprint arXiv:1606.07356*.

Agrawal, A., Batra, D., Parikh, D., & Kembhavi, A. (2018). Don't just assume; look and answer: Overcoming priors for visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 4971–4980).

Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Zitnick, C. L., & Parikh, D. (2015). VQA: Visual Question Answering. In *International Conference on Computer Vision (ICCV)*.

Arik, S. O., Chrzanowski, M., Coates, A., Diamos, G., Gibiansky, A., Kang, Y., Li, X., Miller, J., Ng, A., Raiman, J., Sengupta, S., & Shoeybi, M. (2017). Deep voice: Real-time neural text-to-speech.

Austin, J. L. & Warnock, G. J. (1962). *Sense and sensibilia*, volume 83. Clarendon Press Oxford.

Barsalou, L. W. et al. (1999). Perceptual symbol systems. *Behavioral and brain sciences*, 22(4), 577–660.

Black, M. (1937). Vagueness. an exercise in logical analysis. *Philosophy of science*, 4(4), 427–455.

Cai, P., Indhumathi, C., Cai, Y., Zheng, J., Gong, Y., Lim, T. S., & Wong, P. (2014). Collision detection using axis aligned bounding boxes. In *Simulations, Serious Games and Their Applications* (pp. 1–14). Springer.

Carnap, R. (1955). Meaning and synonymy in natural languages. *Philosophical studies*, 6(3), 33–47.

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

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation.

Das, A., Datta, S., Gkioxari, G., Lee, S., Parikh, D., & Batra, D. (2018). Embodied Question Answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Das, A., Kottur, S., Gupta, K., Singh, A., Yadav, D., Moura, J. M. F., Parikh, D., & Batra, D. (2017). Visual dialog.

Dissanayake, M., Newman, P., Clark, S., Durrant-Whyte, H., & Csorba, M. (2001). A solution to the simultaneous localization and map building (slam) problem. *IEEE Transactions on Robotics and Automation*, 17(3), 229–241.

Dobnik, S. (2009). *Teaching mobile robots to use spatial words*. PhD thesis, University of Oxford.

Fang, H., Gupta, S., Iandola, F., Srivastava, R., Deng, L., Dollár, P., Gao, J., He, X., Mitchell, M., Platt, J. C., Zitnick, C. L., & Zweig, G. (2015). From captions to visual concepts and back.

Fisher, P. (2000). Sorites paradox and vague geographies. *Fuzzy sets and systems*, 113(1), 7–18.

Fong, T., Nourbakhsh, I., & Dautenhahn, K. (2003). A survey of socially interactive robots. *Robotics and autonomous systems*, 42(3-4), 143–166.

Fukui, A., Park, D. H., Yang, D., Rohrbach, A., Darrell, T., & Rohrbach, M. (2016). Multimodal compact bilinear pooling for visual question answering and visual grounding. *arXiv preprint arXiv:1606.01847*.

Gao, H., Mao, J., Zhou, J., Huang, Z., Wang, L., & Xu, W. (2015). Are you talking to a machine? dataset and methods for multilingual image question answering.

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2016). Region-based convolutional networks for accurate object detection and segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(1), 142–158.

Goyal, Y., Khot, T., Summers-Stay, D., Batra, D., & Parikh, D. (2017). Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 6904–6913).

Grisetti, G., Kümmerle, R., Stachniss, C., & Burgard, W. (2010). A tutorial on graph-based slam. *IEEE Intelligent Transportation Systems Magazine*, 2(4), 31–43.

Hariharan, B., Arbeláez, P., Girshick, R., & Malik, J. (2015). Hypercolumns for object segmentation and fine-grained localization.

Harnad, S. (1990). The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42(1-3), 335–346.

Hussein, A., Gaber, M. M., Elyan, E., & Jayne, C. (2017). Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2), 1–35.

Johnson, J., Hariharan, B., Van Der Maaten, L., Fei-Fei, L., Lawrence Zitnick, C., & Girshick, R. (2017). Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2901–2910).

Kafle, K., Yousefhussien, M., & Kanan, C. (2017). Data augmentation for visual question answering. In *Proceedings of the 10th International Conference on Natural Language Generation* (pp. 198–202).

Kennedy, C. (2007). Vagueness and grammar: The semantics of relative and absolute gradable adjectives. *Linguistics and philosophy*, 30(1), 1–45.

Krishna, R., Zhu, Y., Groth, O., Johnson, J., Hata, K., Kravitz, J., Chen, S., Kalantidis, Y., Li, L.-J., Shamma, D. A., Bernstein, M. S., & Li, F.-F. (2016). Visual genome: Connecting language and vision using crowdsourced dense image annotations.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12 (pp. 1097–1105). Red Hook, NY, USA: Curran Associates Inc.

Kruijff, G.-J. M., Zender, H., Jensfelt, P., & Christensen, H. I. (2007). Situated dialogue and spatial organization: What, where… and why? *International Journal of Advanced Robotic Systems*, 4(1), 16.

Lakoff, G. & Johnson, M. (2008). *Metaphors we live by*. University of Chicago press.

Larsson, S. (2015). Formal semantics for perceptual classification. *Journal of logic and computation*, 25(2), 335–369.

Lauria, S., Bugmann, G., Kyriacou, T., Bos, J., & Klein, A. (2001). Training personal robots using natural language instruction. *IEEE Intelligent systems*, 16(5), 38–45.

Liang, M. & Hu, X. (2015). Recurrent convolutional neural network for object recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3367–3375).

Lin, D., Fidler, S., Kong, C., & Urtasun, R. (2014). Visual semantic search: Retrieving videos via complex textual queries. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, (pp. 2657–2664).

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., & Dollár, P. (2015). Microsoft coco: Common objects in context.

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 3431–3440).

Lowe, D. (1999). Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2 (pp. 1150–1157 vol.2).

Matuszek, C., FitzGerald, N., Zettlemoyer, L., Bo, L., & Fox, D. (2012). A joint model of language and perception for grounded attribute learning.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*.

Mooney, R. J. (2008). Learning to connect language and perception. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI)* (pp. 1598–1601). Chicago, IL. Senior Member Paper.

Nilsson, N. J. (2007). The physical symbol system hypothesis: Status and prospects. *50 Years of Artificial Intelligence*, (pp. 9–17).

Quine, W. V. O. (2011). *Two dogmas of empiricism*. Princeton University Press.

Raffman, D. (1996). Vagueness and context-relativity. *Philosophical Studies: An International Journal for Philosophy in the Analytic Tradition*, 81(2/3), 175–192.

Ray, A., Sikka, K., Divakaran, A., Lee, S., & Burachas, G. (2019). Sunny and dark outside?! improving answer consistency in vqa through entailed question generation. *arXiv preprint arXiv:1909.04696*.

Regier, T. (1996). *The human semantic potential: Spatial language and constrained connectionism*. MIT Press.

Ren, M., Kiros, R., & Zemel, R. (2015a). Exploring models and data for image question answering.

Ren, S., He, K., Girshick, R. B., & Sun, J. (2015b). Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39, 1137–1149.

Roy, D. (2005). Semiotic schemas: A framework for grounding language in action and perception. *Artificial Intelligence*, 167(1-2), 170–205.

Russell, B. (1923). Vagueness. *The Australasian Journal of Psychology and Philosophy*, 1(2), 84–92.

Russell, S. J. & Norvig, P. (1995). Artificial intelligence: A modern approach.

Sainsbury, R. M. (2009). *Paradoxes*. Cambridge University Press.

Savva, M., Kadian, A., Maksymets, O., Zhao, Y., Wijmans, E., Jain, B., Straub, J., Liu, J., Koltun, V., Malik, J., Parikh, D., & Batra, D. (2019). Habitat: A Platform for Embodied AI Research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Selvaraju, R. R., Tendulkar, P., Parikh, D., Horvitz, E., Ribeiro, M. T., Nushi, B., & Kamar, E. (2020). Squinting at vqa models: Introspecting vqa models with sub-questions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 10003–10011).

Shah, M., Chen, X., Rohrbach, M., & Parikh, D. (2019). Cycle-consistency for robust visual question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 6649–6658).

Silver, D., Bagnell, J., & Stentz, A. (2008). High performance outdoor navigation from overhead data using imitation learning. *Robotics: Science and Systems IV, Zurich, Switzerland*, 1.

Simonyan, K. & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Skocaj, D., Kristan, M., Vrecko, A., Mahnic, M., Janícek, M., Kruijff, G., Hanheide, M., Hawes, N., Keller, T., Zillich, M., & Zhou, K. (2011). A system for interactive learning in dialogue with a tutor. *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, (pp. 3387–3394).

Smith, L. & Gasser, M. (2005). The development of embodied cognition: Six lessons from babies. *Artificial life*, 11(1-2), 13–29.

Straub, J., Whelan, T., Ma, L., Chen, Y., Wijmans, E., Green, S., Engel, J. J., Mur-Artal, R., Ren, C., Verma, S., et al. (2019). The replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*.

Strubell, E., Ganesh, A., & McCallum, A. (2019). Energy and policy considerations for deep learning in nlp.

Szot, A., Clegg, A., Undersander, E., Wijmans, E., Zhao, Y., Turner, J., Maestre, N., Mukadam, M., Chaplot, D., Maksymets, O., Gokaslan, A., Vondrus, V., Dharur, S., Meier, F., Galuba, W., Chang, A., Kira, Z., Koltun, V., Malik, J., Savva, M., & Batra, D. (2021). Habitat 2.0: Training home assistants to rearrange their habitat. *arXiv preprint arXiv:2106.14405*.

Turney, P. D. & Pantel, P. (2010). From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37, 141–188.

Vinyals, O., Toshev, A., Bengio, S., & Erhan, D. (2015). Show and tell: A neural image caption generator.

Wang, H., Zhang, Y., Yu, X., & Solari, F. (2020). An overview of image caption generation methods. *Intell. Neuroscience*, 2020.

Wijmans, E., Datta, S., Maksymets, O., Das, A., Gkioxari, G., Lee, S., Essa, I., Parikh, D., & Batra, D. (2019). Embodied Question Answering in Photorealistic Environments with Point Cloud Perception. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Williamson, T. (2002). *Vagueness*. Routledge.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, L., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J. R., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., & Dean, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *ArXiv*, abs/1609.08144.

Xia, F., Zamir, A. R., He, Z., Sax, A., Malik, J., & Savarese, S. (2018). Gibson env: Real-world perception for embodied agents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 9068–9079).

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., Zemel, R., & Bengio, Y. (2016). Show, attend and tell: Neural image caption generation with visual attention.

Yu, L., Park, E., Berg, A. C., & Berg, T. L. (2015). Visual madlibs: Fill in the blank image generation and question answering.

Zhang, C., Platt, J., & Viola, P. (2005). Multiple instance boosting for object detection. *Advances in neural information processing systems*, 18, 1417–1424.

Zhang, L., Wei, L., Shen, P., Wei, W., Zhu, G., & Song, J. (2018). Semantic slam based on object detection and improved octomap. *IEEE Access*, 6, 75545–75559.

Zhang, P., Goyal, Y., Summers-Stay, D., Batra, D., & Parikh, D. (2016). Yin and yang: Balancing and answering binary visual questions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 5014–5022).

Zhu, Y., Fathi, A., & Fei-Fei, L. (2014). Reasoning about object affordances in a knowledge base representation. In *European conference on computer vision* (pp. 408–424).: Springer.

Zhu, Y., Groth, O., Bernstein, M., & Fei-Fei, L. (2016). Visual7w: Grounded question answering in images.

Zhu, Y., Zhang, C., Ré, C., & Fei-Fei, L. (2015). Building a large-scale multimodal knowledge base system for answering visual queries. *arXiv preprint arXiv:1507.05670*.

Zitnick, C. L., Parikh, D., & Vanderwende, L. (2013). Learning the visual interpretation of sentences. In *2013 IEEE International Conference on Computer Vision* (pp. 1681–1688).

Appendices

A Datasets

The 3D Scenes and the QA dataset mentioned in Das et al. (2018), are called SUNCG(3D houses) and "EQA V1" (QA). The EQA V1 is a synthetic dataset generated automatically, and constructed based on the setting of the 3D houses in SUNCG. SUNCG is no longer available. Das et al. (2018) changed the SUNCG 3D setting to MatterPort 3D (MP3D). MatterPort 3D is a reconstruction of 3D houses in (SUNCG) scene dataset. The latter also implies that the inital "EQA V1" is not applicable for MP3D.

The new QA dataset for Matterport 3D is available but not the code that generated it. The EQA-MP3D is also a synthetic dataset generated automatically and can be downloaded from Habitat-lab GitHub repository [17]. For generating questions for SUNCG, a code published at this reference[18]. However, there is no code for generating QA for MP3D.

A few of the differences between the question dataset for SUNCG (EQA-SUNCG) and MP3D(EQA-MP3d) are mentioned in Wijmans et al. (2019). However, not in all the information in Wijmans et al. (2019) seems to match with EQA-MP3D that we have. In Wijmans et al. (2019) page(4) it is stated that the number of scene used from MP3D is 76. The dataset we downloaded from "facebookai/habitat" repo on github uses a total 67 scene of 90 scenes available in MatterPort3D. 57 of the 67 scenes are used for questions in the train-set and 10 in the the enviroment. Note that the latter implies that the robot is tested on different scenes from the scenes it has been trained in.

A.1 List of Textual References with Number of Answer Choices

We conduct the analysis on the data by categorizing the questions into references. A "reference", in this example, is a category of typical question-string that can refer to a specific entity in a specific or non-specific space. For example, 'sofa' in questions like "what color is the sofa?" is one reference type. "sofa in the living room," as in "what color is the sofa in the living room?" (color_room), is a second reference type. "sofa in the bedroom" as "what color is the sofa in the bedroom?" would be a third and different reference. In order to gain insight into the data, we categorize the strings of the color questions by reference type. Each textual reference would consist of N number of typical questions. After categorizing all the questions into reference types, we collect the number of answer choices found them as elements of a reference type.[19]

[17] https://github.com/facebookresearch/habitat-lab
[18] https://github.com/facebookresearch/EmbodiedQA
[19] Link to the statistical analysis of the data in a notebook `https://github.com/Al-arug/EQA`.

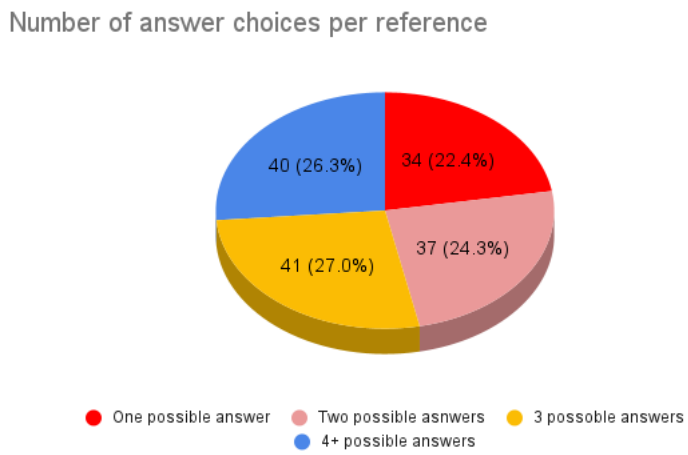


Figure 29: The color questions in the training data sorted by textual reference. In total we find in all color questions 153 references. 22.4 percent of the references have one color answer as the only choice.

.

We find that 22.4 percent of the references have only one answer type, as seen in figure 29.

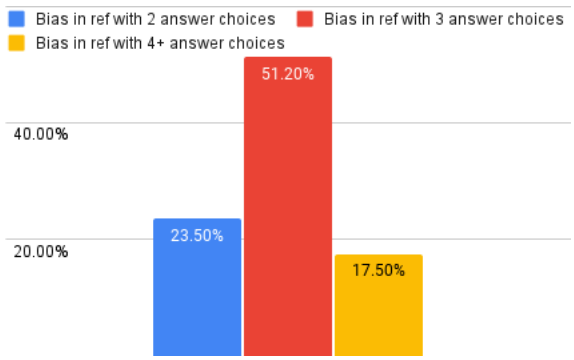
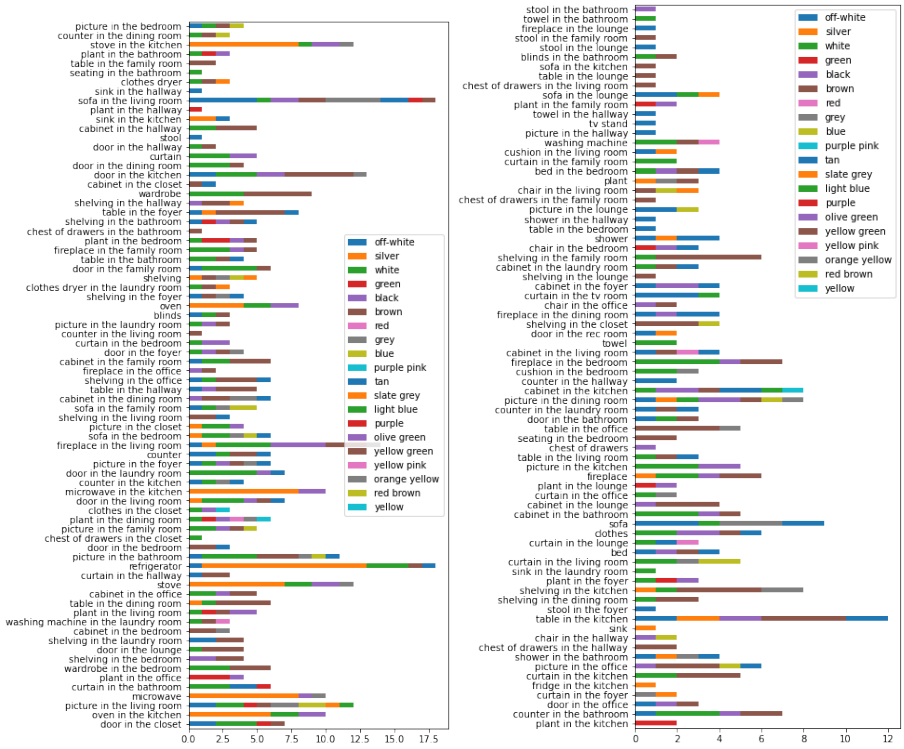


Figure 30

The references with multiple answer choices . In Figure 30 we categorize references per answer-choice. The reference categories are references with two answer choices in one category, references with three answer choices in one category, and references with four answer choices in a different category. The bias for each category is determined differently. In the references with category two answers, a reference is considered to contain biased QA if the answer in 75% of the instances of the answer is the same. For the categories, three answer choices and four answer choices, biased is considered if one color made up 50 percent of the answers in each reference. In total, we get that 23.5% of the references with two answer choices are biased. 51.2% of the references with three answer choices are biased, and 17.5% of the references with four or more possible answers are biased.



(a) (b)

Figure 31: Each row is a textual reference. The number of answer choices for each textual reference is represented by the colorful blocks. One block = 1 answer choice and so on. The colors of the bars are not representative of the named color answer so they should be disragarded.

.

B Habitat-Lab-(EQA evaluation)

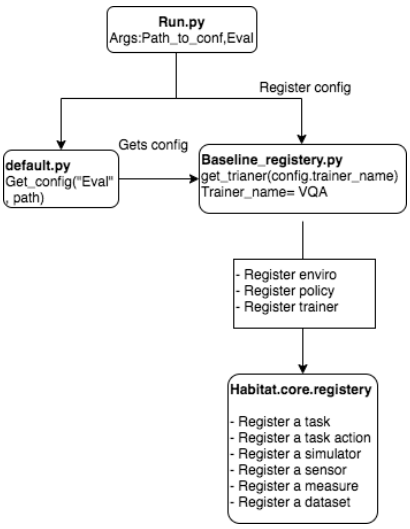


Figure 32: Example of Habitat lab processing the configurations to implement validation for the VQA model.

Figure 32 resembles a map of the code structure when the habitat lab module is initiated to preform validation task for the VQA. Each task has its own configurations and in this example the task is 'VQA evaluation'. As seen in Figure 32, the module takes hierarchical steps in which each step is executed in accordance to the configuration of the given task. In the most down box of the structure we see parts of the commands directed for the simulator, such as insinuating an environment and sensors in the agent. Other commands include registering a data-set which takes part in lab module.

The configurations are processed into commands in Habitat-lab before being passed to the simulator. Habitat lab is the second core component of the system. In addition to giving commands to the simulator, the Habitat Lab module acts as a pipeline that prepares the data-set of the corresponding task. The habitat-lab module,in other words, is the coordinator that informs the simulator of the required setting, and the data loader and processor that prepares the data for either training or testing.

C Generating Spacial Questions

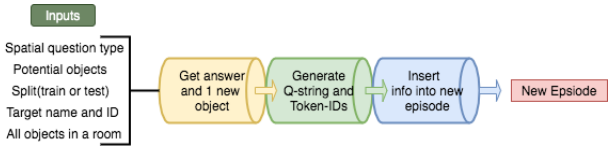


Figure 33: Structure of spatial questions generator.

All the inputs seen in Figure 33 are required to generate an answer from a function called "GetSpatialAnswer". All objects in a room are needed for generating "no" answer. In case of generating a "no" answer the "GetSpatialAnswer" function picks a random object fro the houses that is not in the room. The reason of excluding objects in the room from the selection of a random object for a negative QA is to help us in the validation process, such as we would know if the robot answer 'yes' to a QA with 'no' as ground truth that it is due to bias rather than he robot recognizing the object in the scene.

A selected object of potential objects and the type of spatial question are required arguments for generating spatial question string and token ids.

The last part in blue is conditioned by the type of answer, if it is 'yes' or 'no'. If the answer is yes, geometric information of the target object's pair is passed to it to insert it in the episode. If the answer is 'no', no additional information is added to the episode beside the information of the target object found at the end of the shortest path.

D What is Vagueness?-Philosophical Discussion on the Vagueness of Gradable Expressions

The description of sizes is paradoxical. A well known paradox in philosophy, the sorites paradox, uses the description of a "pile of sand" to display the dilemma of describing the size of an entity. The paradox, described by Fisher (2000),is stated as such: is a grain of sand a pile of sand? No. does adding one or 5 or 20 grains make a pile of the sand?, the answer is still no. We can make inference that adding grains of sand does not make a pile. However, this inference is inaccurate because we might conclude that adding 10 million grains does not make a pile of sand, which is false. This is the reason why it is a paradox. A paradox is when the premises entail a logical inference but a conclusion we draw is false (Fisher (2000), Sainsbury (2009)).

Sorties paradox can be expressed more clearly in prepositional logic, *modus ponens*. *Modes ponent* is a form deductive argument for making an inference. Its rule is based on conditionality of the truthfulness of

a statement, if P is true then Q is true. The inference we make from making a pile is that, if "one grain of sand is no pile" is true, then "2 grains of sand is no pile" is also true. We denote the predicate 'no pile' as *P* and a grain of sand as *a* and the number of grains as *n*; one grain of sand makes no pile is the denoted as such P_{a_1}, "two grains of sand is no pile", P_{a_2} , and our conclusion that adding any number of sands is no pile would be $P_{a_{n+}}$. The process in which we made the inference ($P_{a_{n+1}}$) adding more grains makes no pile is represented as such:

$$P_{a_1} \rightarrow P_{a_2} \rightarrow P_{a_3} \rightarrow P_{a_4} .... \rightarrow P_{a_{n+1}}$$

if one grain of sand is no pile then two grains of sand is no pile, if two grains is no pile then three grains no pile, if three grains is no pile then four grains is no pile, then we make the inference that adding any number of grains is no pile. The conclusion is any number of grains do not make a pile.

Paradoxical predicates are vague form of knowledge. The sorties paradox applies to all the adjectives and forms of expressions that lack a precise form of logical construct; Small, big, bold and expensive are examples of predicates equally paradoxical as the "pile" of sand Kennedy (2007). From an epistemic approach, some philosophers disregard these forms of language expressions as valid arguments/knowledge about the world. In epistemology the questions in interest are how do we know what we know?, in what way was a certain knowledge obtained?. For example, how do we know that a pile of sand is a pile of sand or that a chair is a big chair not a small one; note that the knowledge of 'big chair' or 'pile of sand' is here the 'meaning' of them, what does the symbol "big chair" mean.Williamson (2002) notes that vague predicts distinguished from other predicates by the absence of a precise/clear boundary lines that separates their meaning from its negation, for example, the boundary line of when a pile is a pile and when it is not, or the boundary between pretty and ugly, big and small. On the other hand The distinction between an animal and a tree has a clear borderline, one is inmate and the other is not.

Paradoxical expressions are rejected, in the epistemic view, based on the unfounded precise reasoning that logically defines them. They are considered vague primarily for the absence of higher meaning of measure. The induction to explain what a 'pile' means (in terms of properties of grains) proved to be paradoxical in classic logic, as seen in a previous example. The missing boundary of vagueness in classic logic is the distinguishing line of their truth-value, the one that separates True from false.

Williamson (2002) quotes J.L Austin (Austin & Warnock (1962)) on an argument regarding the usage of expressions such as 'accurate','precise', where J.L Austin disputes ,'If I measure a banana with a ruler, I may find it to be precisely 5 5/8 inches long. If I measure my ruler with bananas, I may find it to be exactly six bananas long, though I couldn't claim any great precision for my method of measurement'. One might say that our measurement of the banana is very precise, but to measure the truthful validity of our measurement is as accurate as measuring it with bananas. In the discussion found in (Williamson (2002)), vague expressions are seen in the epistemic view as a type of ignorance.

It is important to stress that 'vague' refers to the representation of the entity (the description) not the nature of it. Nature is logical, the world either exists or does not exist, the 'pile' exists regardless the vagueness of our measurement of its existence. Williamson (2002) on Bertrand Russel's reference to vagueness, in (Russell (1923)), notes that Russel considers the issue of vagueness as a problem of symbol representation. Quine (2011) refers to the distinction between the symbol representation and the symbol referred to in the world as intention and extension. Intention is the expression/representation/language we use to refer to an existent entity in the world, extension is the entity with its inherited property in the world as it is. Crescent/full moon are intentions, the extension is the same moon object as it is in the world. Quine (2011) elaborates on Russel's assertion to vagueness as an issue with the intentions we use to describe the extension. Williamson (2002) points out the Russel means representation also in extra-linguistic properties, such as the perceptual representation we have of the extension.

Sorties paradox reveal vague expressions of an ordinary language through a logical system of an ideal language. Williamson (2002) on Russel's assertion, in (Russell, 1923), that the existence of vague expressions indicates that language as a whole is vague. An ideal language must have a logical representational unit for every intention-extension Quine (2011). Russell's then invalidates classic logic as suitable to express ordinary language (Ordinary language has vague, general,ambiguous expressions). In the debate whether logic could be used to explain vague expressions, (Williamson, 2002) cites Max Black (Black, 1937), responding to Russel's invalidation of logic in regard to vagueness, that Russel's claim is an evasion of responsibility to describe natural language in systematic way. Black goes on in explaining that the incoherence of vague expressions in logic stems from a trouble that occurs by trying to logically formulate an expression of degree of truthiness without having explicit degrees of truth. For Black, some expressions in ordinary language cannot be confined to absolute truthiness or falseness, they instead could be neither true or false. For example, we can say a person is not tall but somewhat tall, or that adding 3 grains don't make a pile but adding thousands of them could, so 'adding grains of sand makes a pile' is neither false or true but partially true.

Supervaluation logic and solutions for expressing vagueness. Super valuation suggests a three valued truthiness. An expression can be either super-true, super-false or undefined. For a statement to be super-true or super-false a condition must apply to all of its valuations; valuation can be, for example, the *n* number of grains. A statement has "undefined" truth value if it has true valuations and false valuations. This three valued logic allows to distinguish precise expressions from 'borderline cases'(vague expressions). The obtaining of truth value of each valuation of the predicate is referred to as precisification. Precisification are referred to using quantifiers, existential quantifier $\exists$ to refer to a single valuation , and universal quantifiers $\forall$ all the valuations. The logical statement below denotes that there exists an *n* that's a border line for 'is no pile' where *n+1* is a pile. The statement draws a dividing line where at *n*th grain, every *n+1* after the line "is a pile" $\neg P(n_{+1})$

$$\exists n(P(n) \ \& \ \neg P(n_{+1}) \ )$$

Despite the distinction of borderlines in super-valuations, a deficiency in meaning, referred to as highr-order vagueness, still appears. If the borderline *n*th , for example, is 8,000,000 grains of sand where all *n+1* "is a pile", but isn't 8,999,99 also a pile?. This a sort of paradox that found initially. Also the difference between very tall and tall would be still vague, the two predicates would fall after the dividing line with undefined difference.

Fuzzy logic and fuzzy sets proposes a solution of truth range that lies between [0,1]. This range of true values allows to map gradable matters, such as the difference between very short, tall, and very tall. Fuzzy sets maintains a feature of classical logic, that the absolute truth still lies in exclusion of a middle, [0,1],false or true. The range of turthness can be expressed in set theory using notions of intersection, union, complementation and subset which correspond to conjunction, disjunction and negation. In set theory we can denote that a 'definitely is a pile' is distinguished from 'is pile' in a way that 'definitely is a pile' is a subset of 'is pile' without claiming that a pile is a subset of definitely is a pile'. For example, A is a set for 'definitely pile', and B is the set of 'is a pile', the relation can be expressed as $A \subset B$ (A is a proper subset of B), where all the elements in A are in B but B has more elements that could be for example, "is kind of a pile".

Sets can be formed in Boolean functions with conditions which makes it very flexible for usage in computer applications and programming languages. (Williamson (2002)) asserts that the development of fuzzy logic out of fuzzy sets allows for a replacement of [0,1] to any established range of number, for example we could say that *n* is a member of the 'is no pile' set (C) if its value is 0 or more and less than 10,000: n $\in$C If 0$\leq$ n< 10,00.

Vague Descriptions of Sizes For size questions,the object is considered an element of a size set if its within the range assigned for the set. The range and borderlines of each set are determined in relation to a context. Establishing a range for a vague expression relative to a context is defined by (Raffman (1996) ,pp 181).

Establishing a range for sets attempt to reduce vagueness following the semantic conventions for dealing with vagueness. However, even if semantics manage to find logical means to express vagueness as clear as possible, in an epistemic view this clarity is still in question. Semantically, perhaps the burden of clarifying vagueness in natural language is cleared out if boundaries are established with some systematic logical construct. Carnap (1955) refers to vagueness as *intentional vagueness* if no specifications (boundaries) made in the intention. For Carnap, if specifications are made in the intention and vagueness occur then vagueness here is extensional; They mean that vagueness is in the extension itself rather than only a matter of vague expression in the linguistic meaning (intention). An example that higher-order-vagueness remains despite the specification of boundaries, as mentioned before, the pile of sand would perceptually seem the same if it is one unit above or below the borderline of range.

However, for Russel, vagueness would probably remain as an issue of representation. Even if the intention(meaning) is constructed measurably in accordance to some logical relation to the extension, for Russel vagueness still remains in the other types of representation. As mentioned earlier, Russell views ”representation” as including also a photographic symbol such as the perceptual imagery we have of an object. We could assume an object to be of a big size relative to its context, and the ”intention” would seem matching the contextual size if we view the object from a certain point. But what if we go further away from the object ? The object would seem smaller and smaller from the point where we described it as big. The previous point sums the main argument for Russell's view on vagueness as an issue of representation. However, in this regard, he considers vagueness as natural phenomenon or attributed to what he refers to as the 'law of physics'. According to Williamson (2002) Russel refers to the law of physics as 'the appearances of a thing at different places are less and less differentiated as we get further away from the thing'(p.68).