Thesis Title

Institution Name

university.jpg

Author Name

Day Month Year

Contents

Abstract

Our work extends a dataset for Embodied-Question-Answering. Embodied question answering is the task of asking a robot a question about objects in a 3D environment, where the agent is expected to navigate the environment and find the entities in question and answer. The answer system consists of navigation and VQA components. Each question in the dataset is an executable function that could be run in the environment to yield an answer. The published dataset for EQA is EQA-V1, and it is a limited dataset that includes only two types of questions, color and location questions. We use the navigational data, required for training the system, from EQA-V1 and generate new questions of two more types, size and spatial questions. Our data extension is intended to better train the system and enhance its ability in performing the task.

Chapter 1

Introduction

An intelligent robot must be able to understand and resolve references in its environment ([RN95]). Our human ability to interact in reference to our visual surroundings, manifested in language, stems from faculties such as perception and memory([Reg96]). Perception, in particular, is central to our physical experience of the world ([Bar+99]). We conceive the physical world through perception; and we express our conceptualization of the perceptual experience in words([LJ08]). Therefore, the exhibition of intelligent behaviour is necessitated by having a notion of meaning that associates 'words' with the visual/physical world([Nil07]).

In computers we attempt to form a visio-linguistic meaning representation by training systems on connecting low-level perceptual data with high-level meaning(language). 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 system problem"([Har90]).

The VQA model of the EQA task, a CNN-LSTM architecture, shows a lack of visual grounding. In an experiment we conducted on the VQA model, we observed that the robot tends to answer questions relying mainly on the questions (bias). The evidence to the latter appeared when we asked the system questions about objects non-existent in the scene and it answered correctly. For example, we ask the system about the color of the table in the living room and input a picture of a bathroom without a table in it. The questions with the misinforming scenes were asked in a validation round. The results showed an increase in the total accuracy of the predictions despite the absence of the required visual information.

Our hypothesis about an underlying reason for the problem is that the limitations of the dataset played a vital role in the model's learning shortcomings. We find that the dataset is simplistic as it only includes two types of questions. By analysing the existent "color" questions, we observed a bias in the question-answers manifested by a number of typical ground truth answers for typical questions. The VQA model learns to exploit these biases in the dataset and ignore the visual information. This problem has been noted in multiple VQA researches,([Goy+17], [ABP16],[Zha+16], [Fuk+16]).([ABP16]) counters the problem by creating a balanced VQA dataset with reduced language biases.

We extend the dataset as an experimental solution to teach the VQA model to attend to the visual scenes, and to test its overall performance of the task with the new question types. Our choice to include size and spatial questions is motivated by the theory of "spatial language and spatial cognition"([LJ93]). The theory states that the human first acquisition of linguistic names of objects in the physical world is associated with establishing a geometric representation of what defines them. In particular, the conceptual identification of an object might be defined within a spatial relation to other entities, and the image we mentally construct of a concrete noun of a physical property, may appear in the form of its shape.

Our question-generator consists of two major module-components. The first module has a parser class that does semantic annotation extraction, and a second class that extracts and computes geometric data from the semantic annotation. The estimation of spatial relations between objects is inspired and uses parts of code from the EQA question generator of the SUNCG dataset [1]. The second module is the question-answer generator. The question generator uses textual templates to generate question-strings, and construct the questions into an executable function by inserting navigational coordinates and geometric information into each question-sample.

[1]https://github.com/facebookresearch/EmbodiedQA

We extract the semantic annotations from MatterPort3D using the simulator and the sensor's of the Habitat platform. We store the data in a hierarchical structure, starting from the house-environment, then levels, rooms , and objects. Objects are represented by Id- 's , names and geometric bounding boxes located within a global coordinate system. We use the geometric information of the bounding boxes to calculate values such as, how far a boxe's shape stretches from its center, and how much is its volume.

The spatial relation estimator produces three types of spatial relations. The spatial relations are "next to", "on", and "close to". Each of these spatial relations are determined given a geometric criteria of distances between objects' corners along the three axes (x,y,z). The variance of viewpoints, such as how a spatial relation seems from the global view compared to the viewpoint of a robot when it approaches the position of an object, are considered in the criteria that defines a spatial relation between a pair of objects .

Our question generator generates questions for the two types, size and spatial, in three general stages. First it parses navigational question-samples from EQA V1 and takes out the 'shortest path' and the information about the target object that the path leads to. The second stage is generating question strings and ground truth for each "shortest path". The final step consists of inserting the new question with the corresponding geometric information, and structuring them into an executable function.

For size questions, the ground truth about the size of an object is determined by the size of its volume in comparison to the size of the other objects of its type. The comparison is done by the difference of the target object volume from the standard deviation of the median.The templates we use for size questions are as such: 'how big ¡AUX¿ the ¡OBJ¿?', 'how big ¡AUX¿ the ¡OBJ¿ in the ¡ROOM¿?'.

The spatial questions have a binary 'yes' or 'no' answer. All spatial questions have two objects in their question strings. We fill the geometric information in the question-function in a similar manner. Example of the templates used for generating the spatial question : '¡AUX¿ there ¡ARTICLE¿ ¡OBJ1¿ close to the ¡OBJ¿ in the ¡ROOM¿?',

: '¡AUX¿ there ¡ARTICLE¿ ¡OBJ1¿ on the ¡OBJ¿ in the ¡ROOM¿?
: '¡AUX¿ there ¡ARTICLE¿ ¡OBJ1¿ next to the ¡OBJ¿?

The total number of generated questions are 20,000 spatial questions and 14,000 size questions. We trained the VQA model on the new questions, and there was an observable change in the distribution of predicted answers as seen in confusion matrix. The classification report also showed better predictions comparing to the model's performance in the initial dataset. In the main text of the project we include analysis on the results, and more information about the navigational component.

1.1 Problem specifics(unfinished)

In an experiment we conducted on the VQA model, we noticed that the system tends to answer questions relying mainly on the questions (bias). The question of the experiment is if the answer-prediction would change or still be correct if we ask the system about the color of the table in the living room and input a picture of a bathroom without a table in it. The results showed an increase in the total accuracy of the prediction despite the absence of the required visual information.

Yasmeen expirement ....

These results could indicate different things on the model and its data, but the least it could tell is that the system did not need to rely on the images to answer the questions correctly.

The observation of the results opens question-marks on two major components. The first is on the VQA model, as by why it tends to neglect the visual information and rely mostly on linguistic features. The second is a question on the dataset and its contribution to answering the questions correctly. Nonetheless, if one assumes that the model's ineffectiveness stems from a model-data mismatch, how would the system perform if asked different types of questions than the existing ones. This speculation becomes more relevant given that evaluated questions consist predominantly of one question type, color.

color questions is a topic itself, a list of sources [Mon+17]

1.2 Focus and research questions (unfinished)

We have an evidence that the dataset is simplistic, and contains bias in color question.

. Can the system answer more complex questions. (A usful robot should answer a variety of questions.)

. To what extent does the system use the visual information.

. How would the navigation model preform with new questions.

. Does asking more questions improve the system's reliance on vision in color question.

. Does the inclusion of spatial questions improve the system's learning of computational answers- such as olive-green, dark -blue.

. Would a tranformer-based based attention model improve the the preformance of the vqa model.

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. Social robot's usability is very dependent on its exhibition of human intelligence [FND03]. Hence that correct question-answer prediction does not necessarily indicate the system's ability to reason.

An example from the data presented in the Habitat project requires even fewer abstractions, "what color is the sofa?"; The system would only need to rely on perception answer itself "where is the object," then answer the color question.

A common problem in visual question answering is the over-weighting of linguistic over visual features in the answering. [Fuk+16]. Research has shown that models can superficially perform tasks without learning the underlying reasoning process [**turmsimple**]. In visual question answering, it has been observed that a system cheats its way into answering the questions without taking the reasoning steps that humans would logically take to answer a question. [ABP16],[Zha+16], [Fuk+16].In particular,

In Such cases the system answers correctly by exploiting linguistic biases in the dataset, as it tends to rely primarily on the language model and ignore the visual information [Goy+17]

model learns biases in training and manages to give good results in the testing [Sel+20]. The underlying issue here is that the model answers by memorising 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." [Joh+17]; This learning problem is crucial because it makes it challenging to evaluate the model's improvements[Agr+18].

However, color questions could get more complex as "people employ compositional color descriptions to express meanings not covered by basic terms, such as greenish-blue" [MGP16]. It would be shallow to assume that color questions are simplistic, especially if we expect the system to answer colors beyond the basic color terms like "green" and "red."

1.3 meaning

(Add subsection for meaning in the physical space, and how language and physical space influence each other)

The meaning of words is not a mere psychological phenomena. 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 notion 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), contains a perceptual symbolism(mental understanding to what a chair is used for), and refers to an entity with physical features in the world([Moo08]).

In this triangular definition of meaning, vision has an integral part of meaning in which it represents the physical world to language. To recognize a chair, one should, for example, identify the existence of legs, seats, its sizes and geometric shape in a scene. These properties of the physical reference of a chair are most accurately represented in a visual form. Therefore visual recognition is part of the conceptualization process that form the perceptual symbol of an entity.([Bar+99])

perceptual information, however, is more than just visual information. The properties of an object includes 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 tasted.

To 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 the existence of knowledge about the relations of the features that form an entity.([Bar+99]) refers to the process of forming a symbol as 'componential' or schematic, meaning than a notion of meaning is a scheme that is logically constructed. This relational knowledge can be expressed for example in the way we do prototyping and categorization of entities,([LJ08]). The meaning of fully rotten apple is fully understood when we have the knowledge of the negative aspects of eating it. The knowledge about health implications, and the features such as colors and smell of a rotten apple help us categorizing the rotten apple in the category of rotten food.

Reasoning is possible when we are able to conceptualize the existence of representatives of a symbol and their relations among them.

logical relation that links the different representations.

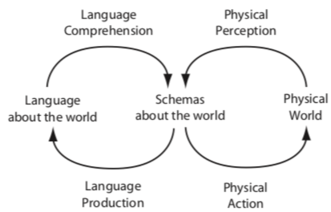


Figure 1.1: Deb Roy

.

In a more concrete example, a reasoning process would "require models to perform inference at multiple levels of abstraction" [Sel+20]. for example, "is the banana ripe?" where it would instantly answer "no". To answer this question, it 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.

s robot do. 1.1 Language is Embedded in t

Distributional representation is a widely used method to represent meanings in words [Mik+13]. Word-meaning in these approaches is defined by its context–The meaning of a word is represented by the word and the words surrounding it. Using language to define language is successful, for example, to make inference that the "university" and "student" are close to each other given their common context of "education".This inferential ability is good for many tasks.

In the field of semantics, researchers seek to represent word-meaning by our mental and perceptual experience of the world. In particular,interactive tasks requires the exhibition of more intelligent behaviour; and this drives the necessity to have more meaningful representations by means of connecting 'words' to the physical word[Nil07]. To get rich meanings by connecting to the physical world would be to connect language with perception[Moo08].

1.4 Grounding meaning

In cognitive semantics, connecting language and perception means connecting low-level perceptual data with high-level meaning(language). low-level data is the representation of perceptional meaning, such as the sensory information from an image. "high-level" is logical inferential.

The ability to connect high with low level representation is important for any task requiring "seeing" and attending answer. "Symbol system problem"[Har90] is when a computational system process a textual and visual input and does not understand the perceptual reference of the text in the image. Grounding text in vision is when we connect the "high-level" symbolic representations such as symbol tokens (word) to a "low-level" non-symbolic representation such as the sensory (visual) features.

Research uses probabilistic models to connect the two domains. Traditionally, the probabilistic learning aims at draw an alignment between sentences, phrases, and words with the corresponding perceptual representations.[Low99].

There are three main approaches to probabilistic combination of perception language. The first is by finding the probability of a grammatical entity(text) being related to a perceptual representation. The second is by classifying each word in a sentence through probability distribution of words over a perceptual representation. The third is classification of word-embedding in a perceptual space.

[Lar17] Categorizes the three methods by their approach to meaning-representations:

1. Meaning as sets 2. Meaning as transparent function 3. Meaning as opaque function

Meaning as sets refers to the methods that use ......

In [Mat+12] [Lar15] we see examples of connecting formal semantics with perception.

[Lar17] evaluates the different methods in respect to compostionality in language. Composotionality is the notion that the whole meaning of a sentence compose of the the independent meaning given by its units (words). Four of the basis of the evaluation are the following:

1. Dealing with intersective compostionality: intersective compositionality is when two words which one is attributed to the meaning of the other "brown bear" where it means a bear that is brown-[brown and bear]

2. Dealing with non-intersective compostionality: non-intersective is one word does not modify the second, such as [Teddy bear]. 'Teddy bear' cannot be mean a bear that is 'Teddy', 'Teddy' is not an attribute of a bear so not [Teddy + bear]. Teddy + bear is instead a different entity with a different perceptual meaning.

3. Learning perceptual meaning: [Lar17] refers to this evaluation measure as amounting to "updating classifiers based on sensory observations of visual scenes and associated linguistic descriptions."

4. Flexibility to work with the state of the art classifiers

The ability to working with the state of the art classifiers gives a flexibility to the model to be improved. Models that classify perception simultaneously is necessary for obvious reasons.

text yo add on why compositonality is important (Human cognition is thought to be compositional in nature: the visual system recognizes multiple aspects of a scene which are combined into shapes [7] and understandings. Likewise, complex linguistic expressions are built from simpler ones [5].SQuINTing at VQA Models:)

The capacity of these models in handling linguistic phenomena such as intersective and non-intersective compostionality is very importnant in many aspects. The better the model at distinguishing the implied meaning in these two forms is good for generating more detailed descriptions for a task like image-captioning.[Nik+19]

The ability to deal with compostionality is even more crucial if we employ these models in interactive tasks, such as dialogue or question-answering. There are examples, mentioned later in this paper, where the vision-language model does not only need to classify object and give answer, but the answer requires reasoning from high-level data (text) and look if the inferential meaning is satisfied within the perceptual representation.

In the evaluation that [Lar17] makes, opaque and transparent functions have the highest points given the criteria mentioned above. According to the assessment, sets as a transparent function is the best fitting alternative, with one minus point on its ability to work with the state of the art classifiers. On The other hand opaque functions scored a '+' on the ability to work with state of the art classifier but scored minus '-' in dealing with non-intersective.

Neural networks (opaque functions) is popular across different vision-language tasks. Due to the flexibility of deep learning methods, defined by its end-to-end applications and ability to be modified with state of the art classifiers, we see the recent systems of multi-modality tasks mostly being neural-networks based.

1.5 image captioning

Image captioning is an extensively researched field where visual-grounding is its center of focus. For extracting ...

Video [Lin+14]

The two main methods are feature extraction and attention mechanisms [Wan+20].

Feature extraction methods can be divided into two main groups. The first group relies on statistical language models(described in the previous section as transparent function). The second group relies on encoder-decoder neural network model that deep extracts features.

1.5.1 Feature extraction-CNN+RNN

Convulutional neural networks are at the core of feature extraction methods. CNN applications, nonetheless, take 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 [LH15] [Gir+16] [Ren+15] , image classification [SZ14] [KSH12],and semantic segmentation [Har+15] [LSD15].

The 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. 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.

Encoder-decoder,CNN+RNN

The encoder-decoder caption generation has a CNN encoder and an RNN decoder.[Vin+15] is an example of an end-to-end neural captioon generation model. In the neural model the CNN process the image features, and the last hidden layer passed to an RNN to generate a description. This method is a sequence modeling that is similar to machine translation. This means that image features are translated into words. The sequence is predicted by finding the probability of a certain description from a corpora given the features of an image.

RNN are known to be used widely in language technology applications. Rnn is used , for example in text-to-speech [Ari+17] and machine-translation [Cho+14],[Wu+16]. 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. Such an incremental-sequence prediction is suitable for sentence predictions in respect to word dependency.

RNNs have a issue of 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 seizes to improve.

Long-Short-Memory network (LSTM) provides a good alternative for avoiding the disappearing gradient. The gradient in RNNS vanishes in long sequences where the gradient keeps reducing. The architecture of the LSTM allows it to keep information stored for very long sequences. The latter gives it the ability to control the values of the gradient by updating it with information stored in the 'forget gate' form previous steps, preventing the gradient from vanishing.

. (check this link for vanishing https://medium.datadriveninvestor.com/how-do-lstm-networks-solve-the-problem-of-vanishing-gradients-a6784971a577)

https://towardsdatascience.com/understanding-and-calculating-the-number-of-parameters-in-convolution-neural-networks-cnns-fc88790d530d

https://builtin.com/data-science/gradient-descent https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484

https://arxiv.org/pdf/1412.2306.pdf

1.5.2 Feature extraction- statistical language model

Statistical language model, as in [Fan+15], generate descriptions in three stages. First it detects words in an image using a convolutional neural network (CNN) for extracting image features. The incorporation of language at this stage happens using multi-instance learning(MIT)[ZPV05]. The second stage, the statistical language model detects the most likely sentence to make of the words from a pre-defined corpus. In the third stage the sentences undergoes a re-ranking stage where the sentences are combined to generate captions.

1.5.3 Attention-mechanisms

In the previous section the discussion on the methods and implementation

1.6 Dialogue and VQA

In this section of the text 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+17] trains a system in encoder-decoder model on a data set of 2 pairs dialogue with an image content.[Sko+11] trains a system on learning concepts with visual content in an interactive-learning approach.

To make a true statement about the computer's capability to engage in a visual dialogue, it must be first ensured that the computer actually understands the questions being asked to it. Otherwise, dialogue is very complex with many elements determining it succession. In a dialogue with visual content, the computer must, furthermore, understand the questions within their visual context. It is reasonable that we see increasing research on "Visual Question Answering" and less on visual dialogue as a whole. Improvements in VQA intuitively means that we are moving closer in the direction of having an interaction with a computer in a visual dialogue.

[Ant+15] 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+15] 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.

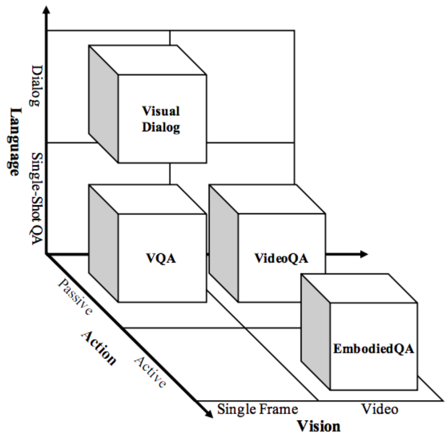
Data-sets that use MS COCO scenes such as [Gao+15], [Yu+15] in addition to [Ant+15] used human workers to write the texts for the scenes. Other data-sets are generated automatically such as [RKZ15].

[Zhu+16] introduces 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 [Kri+16] contains the largest data-set with regions for both VQA and Image-captioning. Object-region approach is intended to improve visual grounding, by marking the regions of the image that the strings refer to.

restricted visual Turing test to evaluate visual understanding. The DAQUAR dataset is the first toy-sized QA benchmark built upon indoor scene RGB-D images. Most of the

1.7 EQA

Embodied Question Answering is one of the latest interactive tasks introduced in the vision-language language field. The EQA task consist of two sub tasks. The first is navigation, and the second is Visual Question Answering. The agent needs to successfully navigate to the target object. Once the agent reaches the goal it would stop at the target location and would process the question and the visual input to answer the question.



. Simultaneous Localization and Map Building(SLAM) is a problem where a robot should be able to map an unknown environment without a GPS or local map. Simultaneous localization is when a robot discover its surrounding and simultaneously construct a map while aware of its changing location. This means that the robot should extract information from its surrounding and learn the map as it goes.[Gri+10] [Dis+01] [Zha+18]

The robot has to discover its surrounding and simultaneously update/know where its location in order for it to construct the map.

Since the embodied question answering is multimodality, related works can vary between different domains. The first category of research includes the embodied-question answering task, including navigation and visual question answering. The second is research that aims to improve the VQA model, which on its own can be considered as multimodality, where the architectures of such systems aim to integrate visual and linguistic features (Images and questions).

Chapter 2

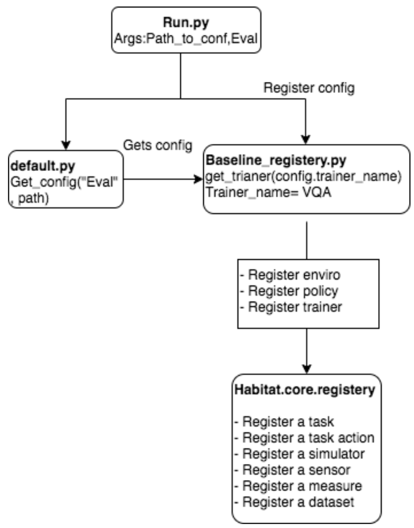
Background

2.1 Habitat overview

The name 'Habitat' is derived from the notion of learning within and from an environment. Imitating our natural habitat, the Habitat Project facilitates spawning an agent in a simulated environments with the possibility of teaching the robot to preform different tasks.

The perquisites needed to test or train an agent for a certain task in a given environment, are facilitated by a core component called Habitat Simulator. Habitat Simulator is responsible for simulating a environment and insinuating a robot in it. The simulator acts depending on the configurations given to it.

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. In other words the habitat-lab module is the coordinator that informs the simulator of the required setting and prepares the data for either training and testing.



Figure(x) resembles a map of the code structure when the habitat lab module is initiated to preform a task. Each task has its own configurations and in this example the task is 'VQA evaluation'. As seen in the figure, 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 dataset which takes part in lab module.

The Habitat Project introduces a new task called Embodied question-answering. The EQA task consist of two sub tasks. The first is navigation, and the second is Visual Question-Answering. The agent needs to successfully navigate to the target object. Once the agent reaches the goal it would stop at the target location and would process the question and the visual input to answer the question.

The system consist of four modules. They are vision, language, navigation and question-answering. Vision and language are used for both navigation and question-answering.The vision-language modalities are fine-tuned differently for navigation and QA.

Navigation and question-answering are trained separately. Imitation learning is used for navigation, and supervised learning for question-language.

There is no available connected model that connects(Nav,VQA). The researches elaborate that the system preform poorly if the two modules put to work together. Both modules use the shortest path as a way to reach the scene of the question. However, the navigation might go off the shortest path and seek to take more actions to reach the goal. This might lead to distorted images and inaccurate view point of the object in question. "Noisy or absent views" would confuse the question-answering model. Therefore, the navigation is frozen when once it completes a navigational episode.

2.2 vision

The vision of the system relies on egocentric 224x224 RGB images processed in CNN. The CNN encoding has the functionality of a "multi-task pixel-to-pixel prediction framework," which consists of 4 5x5 Conv, BatchNorm, ReLU, 2x2 Max-Pool blocks, and they produce a fixed-size representation.

It is possible to train the encoder-decoder on generating three sensory information. The three decoders, which can also be referred to as sensors take the functionality of: 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)."

In the baseline models, different tasks take different sensors.Not all the above-mentioned sensors are used in all the baseline tasks. Since navigation and VQA are trained and evaluated separately, we refer to them as separate "tasks". The two tasks in EQA take the following sensors:

Navigation: "depth" and "RGB". Depth sensor is essential for the agent's capability to navigate. With depth sensor it could estimate distances and avoid colliding with obstacles.

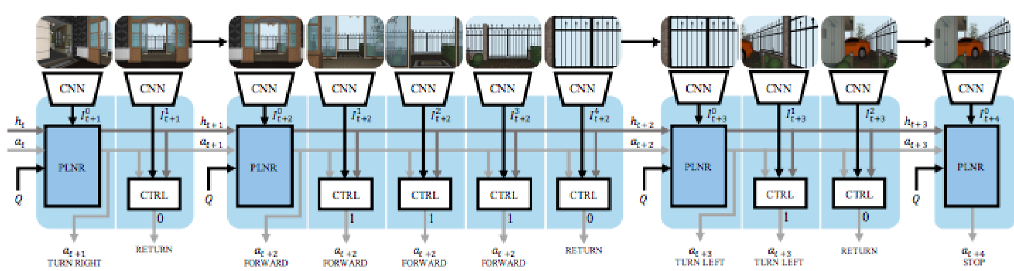
VQA: "RGB". No reason mentioned to why the other sensors are not used in the question-answering baseline module.

"The range of depth values for every pixel lies in the range r0, 1s, and the segmentation is done over 191 classes"(p.11). (page,6)

2.3 language model

The language models is a 2-layer LSTM with 128d hidden layers. The language model's role is to encode the questions and produce a representation. The language-encoder is trained differently for the navigation and VQA. The encoder learns to focus on different strings for each task. Some words in the question can be crucial for executing one of the tasks while less useful for the other task. An example of a question, cited from the EQA paper, " 'What color is the chair in the kitchen?', 'color' is irrelevant for navigation and 'kitchen' matters little for question answering (once in the kitchen) "

2.4 navigation



Habitat's navigation is referred to as PACMAN. It consist 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.

The controller takes the previous hidden state and action-decision and executes the action. A visual input is passed to the control then the controller classify the next decision of two possible decisions. Either to repeat the last action given by the planner or to return to the planner. The controller can repeat the same action maximum five times then it automatically returns to the planner.

Visualization of the navigation is in figure (1). 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 :

"I_t^ndenote 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 $a_t \in \{forward,\ turn-left,\ turn-right,\ stop\}$ "p(6)

For eample the first step-decesion 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 basis of the controller decision is a bit unclear)

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:

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

"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"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",repeats and controls the action in order to align I_t^n with intent given by the planner.

2.4.1 Imitation-learning

2.4.2 Nav evaluation

2.5 VQA

2.6 datasets

The datasets consist of two parts. One is a 3D indoor enviroments, and the other is a question-answering data-set. The 3D enviroments are constructed images that assimilate real indoor enviroments. The 3D Scenes and the QA dataset mentioned in [Das+18], 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+18] 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.v1 is a synthetic dataset generated automatically. There is a available code to generate QA for SUNCG, but the question generator for the latest published QA for MatterPort is not available.

A few of the differences between the question dataset for SUNCG (EQA-SUNCG) and MP3D(EQA-MP3d) are mentioned in [Wij+19]. However, not in all the information in [Wij+19] seems to match with EQA-MP3D that we have. In [Wij+19] page(4) its 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.

We refer to each question-sample in the EQA data-set as an "Episode". Each question is an episode, because the sample contains also, on the topic of the question-info, geometric information and shortest paths. Each episode is applicable for navigation and VQA, and can be run for each task separately.

The EQA-v1 dataset consists of 1950 validation sets and 11000 training questions.

2.6.1 Scene Dataset

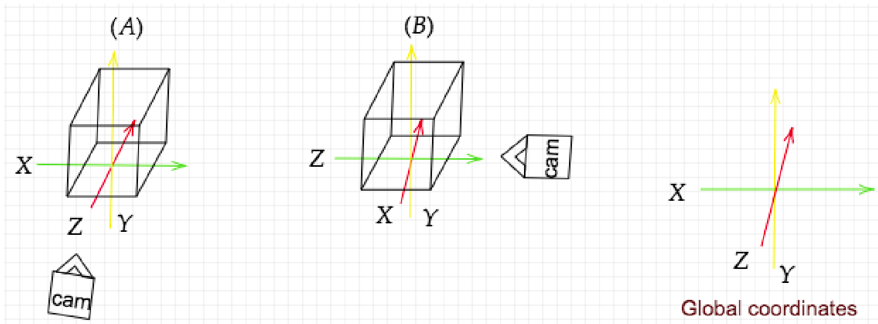
(restructuring is required– more precision) (examples to rephrase– why do we need the location in global coordinates and why the camera views are also important)

The MP3D dataset provide 90 segmented houses with their semantic annotation. The semantic annotation is segmented based on the structure of the house. The segments consists of house levels(floors), to regions(rooms), and objects. The annotation is organized accordingly, such as that an object is annotated and indexed in relation to room and the floor its located in.

For example, 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,obj level2:......]

Each semantic annotation include geometric information. The geometric information consist of elements as 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 [Cha+17](p3). A way to allocate an object is to find its location in a accordance to global coordinates. Let's say 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); and let's say all the objects are spawned through out the house's (x,y,z) axis where each objects location is defined by its distance to the house center. When annotated, the objects are viewed through a camera. The description of their geometric location, thus, should consider the view-postion of the camera.



In graph (A) in figure 3, we see that the camera-view of coordinates align 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, if the camera is positioned to the right of the object from our view, as in graph (B), then we say that the camera view of coordinates is not aligned with the global view. We notice in graph (B) that from the camera view, the "global X" is "Y" and vice versa.

Some geometric calculations cannot be preformed 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 location of an object is referenced by the camera's position then we would get locations relative to the changing position of the camera in a time-span.

To globalize the orientation of the view, measures such as top-down view of a map, or calculating the rotation of the camera 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 environments:

1. World coordinates(global): World coordinates(global): The coordinate system that starts at the center of the world; a house in our example. The center of an object in this coordinate system, is then decided by its distance to the center of the world.

camera-view coordinates: The coordinates from the camera's views. The center of this coordinate system is the position of the camera. The center of the object in this world is defined by its distance to the camera.

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

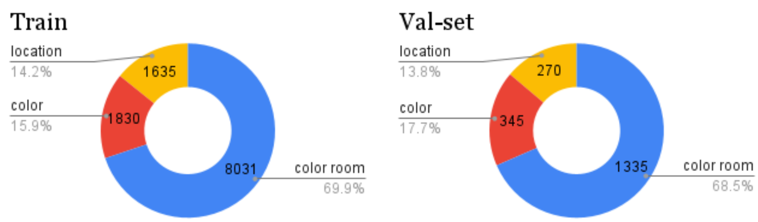
The center of all these views is (0,0,0). We described above that the world coordinate system allows us to measure distance between objects in a world map. The camera view is useful if a robot is expected to navigate an environment and describe spatial relations between objects such as "next to", "above". The local view could tell about the size of an object. In particular, the (x,y,z) from a local point of view tell about how far the object stretches from its center where the center is (0,0,0). The local view can be referred to as "radius".

MatterPort 3D provide the views decribed above. We discuss in more detailed the usage of the object's location in global coordinates and the local view in details in the implementation part.

2.6.2 EQA (Task Dataset)

(More information to include– 1. How they filter out questions based on entropy, and how they filter out objects based on size..2.How many unique question there is. 3. Explain more thoroughly how the singleton(object,room) works)

The question-answer data-set contains three types of questions. Each question in the detest is a function that can be executed in the environment to give an answer. More in section (3.2)



(To include the number of unique questions here) There is a total 11496 questions in the train split and 1950 questions in the val split. As seen in figure (5),in the train split there are 1830 questions "color" type, 8031 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.

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 and look for the object and answer the question. For the agent's to be successful at reaching its target, it needs to know the difference between rooms, and objects, as by implicitly recognizing that a certain room is a living-room, 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 "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 it all it needs to do is to look around.

- location template: "What ¡room¿ is the ¡obj¿ located in".

Querable objects and rooms

The questions ask about 50 unique objects. [Das+18] in (page 4) describe the process of object and room selection for question generation.The following quoted from (page 4):

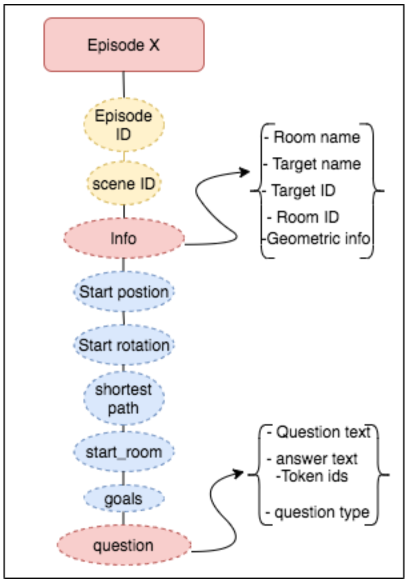
”select(objects)-¿singleton(objects)-¿query(location)”

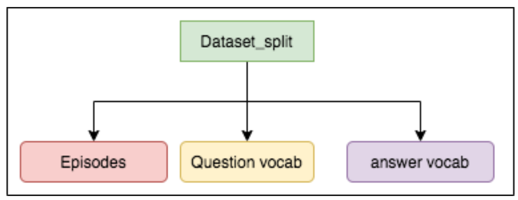
The above represents the steps taken for finding object and location to fill in the questions template. select(objects) is a function that collects all the objects in the house. singleton(objects), filter out an object that occurs only once in the house; query(location) finds the location of the object.However, this applies to the old dataset in SUNCG.

In EQA-MP3D, the object in question is not unique to the house but to the room. The latter means that for an object to be selected for a question, there need 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 happen to be more instances of the same object in the room.

we observe that all the objects that the robot is asked about in testing have occurred in the training questions. While it has been mentioned earlier that the robot is tested in different scenes from the scenes it was trained on, similar objects from the training co-occur in the testing. The latter means, in particular, that the robot is unfamiliar to the test scenes but familiar with all the objects that are being asked about in the test. This information is also stated in [Wij+19].

Structure





In figure (x) we see the top structure of the val and test. *Episodes* refer to each question-function in the 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].

"Question vocab" and "answer vocab" in the "train" and "val" are identical to each other. When using each split of the dataset, the answer-tokens that are considered are the ones contained within the episodes instead of the word-lists mentioned above.

Each question-sample is an episode that consist of multiple layer information. The structure of one episode of all the "episodes" is as seen in figure(x). We describe the elements of an episode in the following:

House ID: The house ID given by the house ids in MatterPort3D. **Episode ID**: The episode index in the range of the split's length. **Info**: This element contains all the information about the the object and room in a question. The information is structured as such:

Information about the traget-object is the first layer within "info":

centroid: The center of the object's box in the global coordinates. Box is the area that labels the object. When the center is globally oriented we would refer to this center and box as Axis-aligned bounding box(AABB), which means that (x,y,z) axis of the center are aligned with global coordinates.

radi: It tells how far the box (object) stretches from its center one direction of each axis. The value of radii is relative to the object itself (from the local view), where the center is zero. If we have, for example a radi of (2,1,4), this means that the object's box stretches +2 and -2 from the center on the x axis. The boundaries of the object's box relative to itself is referred to as object oriented bounding box (OOB).

level: at which level-floor of the house is the object located in.

room-id, *room name*,*obj Id*,*room 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 second layer is information about the room:

Information about the room is similar to the type of information given for the objects. Th information is *floor-level*, *room-id*, *room name*,

Final layer consist of a "question-meta" which includes the color of the object. This section also includes question-entropy .....

The elements that are marked in blue in figure(x) are navigation-related material.

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

rotations: This is the rotations that the agent have to do while navigating. It stands as supplementary information for the shortest path

goals: Goals are the destinations that the agent should reach in navigation. The goals stand for the possible view points from where the the target object can be looked at by the robot. Each view point consist of geometric position and the rotation toward the target object respective to the position.

2.6.3 Bias and answer distribution

Evaluation

Baselines

expirement The idea is to extend the question asked for the agent. The two types of questions are size and spatial. The process of question extension includes using information from the initial EQA-v1 dataset, which consists of color, color-room, and location questions. Each question sample has a target object with corresponded information as object ID, room ID, Scene ID, question(token-ids and text), and shortest path. We pick the object and the room ID for every question sample to extract the rest of the information about the other things in the room. The extracted information is the volumes of the objects and the

Chapter 3

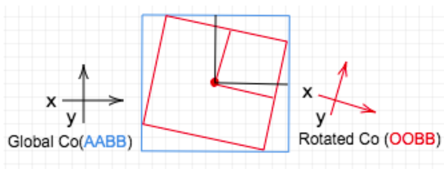
implementation

3.1 Overview Extending Dataset

The idea is to include more questions about the same objects and the scenes found in EQA-v1. The question generation copies the EQA data-set and modify the so called-episodes in it. The episodes, as mentioned in previous sections, are executable functions when inserted in an environment they yield an answer. The scenes denote the visual scene of the destination goal of the a navigational episode. The system learns to reach its navigational goal by a shortest path included in each EQA-V1 episode. Our new questions use the same shortest path found in EQA-V1.

This project consist of two major module-components. The first module is a parser that does data extraction, and acts as a processor for raw data by transforming into usable geometric information for generating question-answers. The second module is the question-answer generation. This chapter describes the projects construct and the usage of each part of it.

3.2 Required data and data processing



The geometric information can be classified into two main categories. In a 3D environment we can imagine each object having a labelling box rotated and oriented around its shape, and other box that bounds the first box with the global coordinates. In figure(X) we see a demonstration of the two boxes in 2d squares. The red box, that is meant to surround an object, is referred to as 'Object Oriented Bounding Box' (OOBB). The coordinates of the OOBB are rotated with rotation of the object (rotated in accordance to the local view). The blue box is referred to as 'Axis Aligned Bounding box'(AABB). The axis of AABB are aligned with global coordinates.

We extract and save one specific information type of each box. The feature we take is the "radii", or else can be referred to as half-extents. The half-extents (radius) can be helpful in representing the object in different ways.

We use the radius of the OOBB box to calculate the size of an object. The volume of OOBB gives more precise estimation of the size of the object, as the box is more enclosed around the object.

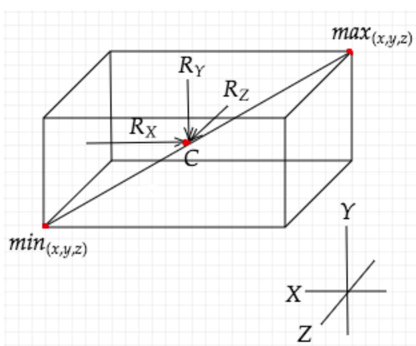
We use the radius of AABB box to measure the distance of an object to other objects. We can measure the distance between objects in an environment by the distance between their centers. More precise measurements would be to measure the edges or the corners of the object. We get the corners by measuring how far the box stretches(given by radius) from the center.

Important to mention that we locate entities on a map with coordinate points that are positioned in accordance to one coordinate system (grounded in a the global map). Otherwise the numbers that represent positions would be in-indicative of points in the global view . It is for the latter reasons, the centers (located globally) are helpful to measuring distance–because they are located on the same coordinate basis.

The AABB box, provide a straightforward estimation of the positions of the object's shape in the global map. The local view of the AABBs are aligned with world coordinates, therefore allocating its corners globally would only require an estimation of how its radius(given in alliance with world coordinates) stretches from the center. True that the OOBB corners are better representatives of the objects corners, however, locating the oobb corners in the world map is not simply done by measuring how its radius stretches from the center in the direction of the global axis as in AABB. The radius of the OOBB is given along its local view axis(its rotation), meanwhile the center point is given in the world axis. Thus, locating the positions of the OOBB, using the same radius-length measure, would require adjusting the radius to the direction of its rotation. Therefore, the global-alignment characteristic of the aabb provides a direct way to locating its edges.

The corners of the OOBB would be the most precise representation of the corners of an object.

For measuring the distance, a readable measurement of distance between entities on a map is necessitated by



We make calculations from the data we extract in order to obtain other necessary info for generating question. The first calculation is finding the 'min' and 'max' of a box given an object's center and half-extents(r). In figure (X) we see visual representations of the extracted and calculated data. The min represents the corner of a box that has the lowest value of (x,y,z) and max is the corner with largest value for (x,y,z). Other way to put it, the min represents the corner in the minus direction from the center in all the axis, and max is the corner on the positive direction from the center in all axis.

(c) We subtract and add the half-extents with the center value. Min =

3.3 First module- Data parser

Our house parser consist of two classes. The first class is a class that parses the houses into a structural data. The second is functional class we use to find near objects close to a target object. The latter class is used

This project include two different methods of annotation extraction. The first source-method is the raw annotations given in the 'house files' of the MP3D data-set. The second method uses Habitat's simulator and sensors. The annotations extracted from the sensors in the Habitat's simulated environments provide more computed information and slightly different raw data MP3D annotations; In particular, some object names are different, but the rest of information, such as object ids and location-centers, is consistent with the annotation of the MP3D.

In the existing generated question-answers data-set we use the data extracted from the Habitat semantic sensors. The main reason for choosing Habitat's semantic sensors is because they provide a computed geometric information of the objects such as the location of an object within an Axis Oriented Bounding Box (We elaborate on this term in the coming section). An additional important reason for this choice of extraction is that some of objects names output-ed by the sensors are aligning with the names found in the original EQA-V1 data-set. For example, object names in MP3D such as l-shaped sofa and rounded-sofa are transformed, in Habitat's sensor, into their Hypernym category 'sofa'. Choosing object names that are aligning with names found EQA-v1, is helpful for having the overall data consistent with each other when we emerge our generated questions with EQA-V1.

The second major component - get close distances

3.3.1 Class$_1 Method1 - Annotations from MP3D files$

In the Matter port annotations, Each house environment comes with three files. The three files are x.house,x.ply and x.. We collect the annotations from the x.house files house.

Each house file comes with eleven line-types of annotations.. The lines are marked by a capital letter as a marker; the first letter-marking to the last letter are as 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 am going to explain only the type of information that we use in this project.

The only data we extract from the house file, is the "O". The "O" lines contain information about the objects in the house. 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]

The data of the object in the line seen above comes in a string form, and 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 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. *a0x a0y a0z a1x a1y a1z* these are the rotation of the OOBB and AABB. *r0 r1 r2* represent the radius of the object from the center on the (x,y,z). Finally the last "0"s in the line have no meaningful value, and therefore are ignored.

We extract two types of raw information from each object's line of annotation. First we take the obj and room indexes (ids). Second is the [px py pz] and where we categorize it as the center of the object's box. Third is the [r0 r1 r2] (radius-half-extent).

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

3.3.2 Class$_1 Method2 - AnnotationsfromHabitat'ssensors$

3.3.3 Second class - Distance and size calculator

The main functionality of this class is to find a spatial relation between pairs of objects in a room. It takes as an argument scene and room id and uses this information to access the objects in a room from the parsed houses files.

It outputs three types of spatial relations between a pair of objects. The pairs spatial relation is specified by whether an object is 'on' , 'next' or in unspecified 'close' distance to a second object. The pairs are organized in a dictionary, one key for each spatial relation. This information is used for generating positive spatial questions.

The spatial relations mentioned above are measured by calculating the distance between the corners of the objects' bounding boxes along certain dimensions. The corners obtained using the 'min' and 'max'. If we iterate over the Min(x,y,z) and Max(x,y,z) we get the other six corners of the box. Figure 3.1 illustrates the eight corners, the view point of the cube is rotated to the right for the sake of viewing all the points in the cube. 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 on 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 .

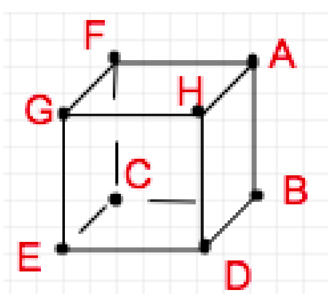


Figure 3.1:

We could get the rest of points from the Min and Max of an AABB for the reason that AABB's are not rotated and aligning with the global view. To express it better, we image the global point of view of the AABBS as a view facing a group of adjusted and not rotated boxes.

For our example in figure 3.1, the values of the six corners of the box found from the Max and Min in addition to the corners of the Min and Max would be as such:

$$A = (x_{max}, Y_{max}, Z_{max}), F = (x_{min}, Y_{max}, Z_{max}), H = (x_{max}, Y_{min}, Z_{max}),$$
$$B = (x_{max}, Y_{max}, Z_{min}), D = (x_{max}, Y_{min}, Z_{min}), C = (x_{min}, Y_{max}, Z_{min}),$$
$$G = (x_{min}, Y_{min}, Z_{max}), E = (x_{min}, Y_{min}, Z_{min})$$

A visual representations of the corners and their values seen in figure.X

The definition of each of the mentioned spatial relation , is an approximation of how we define them as humans. Below we describe how each of the 'on', 'next to', and 'close' relation is determined between the objects.

On

next to

close to

3.4 Second Module - question generation

In order to run the generator, the arguments required are the type of question, path to the val and train splits.

The question generator generates questions of one type at the time. Questions with the string "room" in is considered a different type from a question that refers to objects without a room. For example, the question "How big is the table?" has the type "Size", and the question "How big the table in the living-room?" is of type "Size room". In order to generate size questions with and without reference to a room, therefore, requires ruining

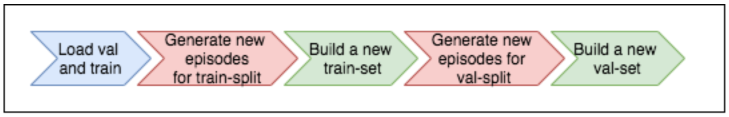


Figure 3.2: Split generator (The top part of the code)

the code, one separate time for each type.

As seen in figure the train and validation splits are generated in two different iterations. The reason for generating the two splits in two different stages is to keep track of the number of question-answers generated for each split. Emerging the two splits and splitting them randomly at the end might create and imbalance between the number of "yes" and "no" answers in each split. The current code controls the distribution of answers in the train and val sets. Otherwise, leaving the type of answers uncontrolled would leave a bias towards one answer over the other; this is particularly important for the reason that binary answers makes it even easier for the system to catch the bias.

The episode iterator is the core component in the code. The first function in the iterator is to extract information, such as the object name and id, scene ID and room ID, from the EQA-V1 episode. The function then branches off into two streams depending on the question type given. The branching consist of two conditions. One stream conditioned for questions of 'size' type and other for questions of "spatial" type.

The generators for size and and spatial types share very similar components. The general idea for generating any question rely on three-steps basis. First to generate a ground-truth answer for the given question, which is the most important stage in the generation process as it requires calculating values from the data in the houses. The second step is to generate a question string. The third step, is a wrapping up stage that transform a question-answer into a complete episode that is executable in the Habitat environment and compatible with training code of EQA. In the coming two sub-section we describe in detail how questions are generated for each type.

The final step in generating

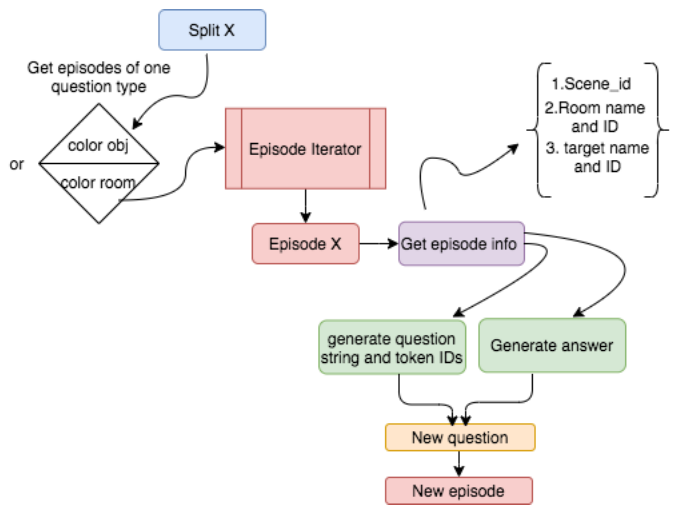


Figure 3.3: Structure of the question generator

We generate two major types of questions. They are Size questions and Spatial questions. Spatial type

There are three answer types for the size questions. Big, Small, and medium. The second

in figure we see three types

3.4.1 size-questions

Size questions are generated through three steps. The first step is generating a ground truth answer about the size of the the target-object found in EQA-V1 episode. There three possible answers are Big, Small, and Medium. The second step is generating a question string and token ids. The final step consist of filling the question-answer in an episode form, with shortest path and the rest of object's info copied from the EQA-v1 episode.

Size answer

The size answer is generated in a function referred to as "GetsizeAnswer". This function takes as an argument the target-object's name and size and returns an answer about its size. The function calculates the size of the target object in a similar way as the rest of the sizes of the objects. Volume of the AABB = W x L x H. The next step in the function is to compare the size of the target to the sizes of the objects of its type. As mentioned earlier, the sizes of all objects are stored by type in a file called "obj-volumes".

The relative size is determined by its deviation from the standard of its type. There are three answer choices, big, small or medium. First, the standard deviation, denoted below, is calculated for each object type. For example, there are n samples of sofa in the 'sofa' object-type

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

The answer is 'small' if the objects' size is smaller the median size of its type minus the standard deviation, 'big' if the size is larger the median + the standard deviation, and middle if the size of the object is within the standard deviation added and subtracted from median.

Size question

und all the houses, including the object the enviroments that are not

The question itself is one string template that is filled

The object Id, scene ID, and object size

The first step requires three elementary information about the target object found in EQA-V1 episode.

3.4.2 spatial-questions

The generator for spatial questions

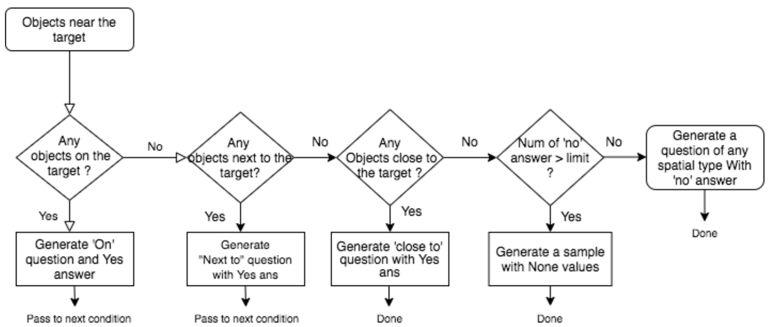


Figure 3.4: Structure of the question generator

3.4.3 Filtering objects

Chapter 4

Training and Evaluation