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

| | |
|---|---|
| Master's Thesis: | 30 credits |
| Programme: | Master's Programme in Language Technology |
| Level: | Advanced level |
| Semester and year: | Spring, 2020 |
| Supervisor | () |
| Examiner | () |
| Report number | (number will be provided by the administrators) |
| Keywords | (Artificial Intelligence, Natural Language Processing, Computer Vision, Visual Question |

## 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.

# Preface

Acknowledgements, etc.

# Contents

# Introduction

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Two citation examples:

Testing Unicode: Göteborgs universitet

*Testing* **testing** TESTING some font series.

Testing a formula:

$$P(X) = \sum_{i=1}^{N} P(A_i)P(X|A_i)$$

Testing a table:

| | |
|---|---|
| cell 1 | cell 2 |
| cell 3 | cell 4 |

Table 1: This is a table.

# Example section (heading level 1)

Text

## Example subsection (heading level 2)

xxx

### Example subsubsection (heading level 3)

xxx

# background

problems with situated agents. Why robotics not in real life.

An intelligent robot must be able to understand and resolve references in its environment (Russell & Norvig (1995)). Our human ability to interact in reference to our visual surroundings, manifested in language, stems from faculties such as perception and memory(Regier (1996)). Perception, in particular, is central to our physical experience of the world (Barsalou et al. (1999)). We conceive the physical world through perception; and we express our conceptualization of the perceptual experience in words(Lakoff & Johnson (2008)). Therefore, the exhibition of intelligent behaviour is necessitated by having a notion of meaning that associates 'words' with the visual/physical world(Nilsson (2007)).

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"(Harnad (1990)).

Situated language with virtual robots,– actions and language. Why this is relevant.

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,(Goyal et al. (2017), Agrawal et al. (2016),Zhang et al. (2016), Fukui et al. (2016)).(Agrawal et al. (2016)) 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"(Landau & Jackendoff (1993)). 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.

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.

Intro:

## 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

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

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 Monroe et al. (2017)

## Focus and research questions ( unfinished)

Intro:

Artificial data-sets, Meaning, grounding, image captioning,

problem:

- limited coverage of questions in the data-set. 2 types - Bias in the color questions. 3 types - Lack of visual grounding. consequential

Research question:

- How can we extend the data-set with more sophisticated and natural questions. - What is the bias formed by color questions. - How does the VQA system preform with the new question type. - Does asking size and spatial question improve the system's attention to vision.

- overview how habitat is constructed

- Does asking more questions of size and spatial types improve the system's attention to vision?

- How does the system preform when we ask more question types?

We have an evidence that the dataset is simplistic, and contains bias in color question. (cause of the problem). problem- lack of visual grounding.

- How does the system preform when we ask more question types? - How does the

questions. (A usful robot should answer a variety of questions.)

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

. Does asking more questions of size and spatial types improve the system's reliance on vision, rather than language.

.

. How would the navigation model preform with new questions.

. 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 Fong et al. (2003). 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. Fukui et al. (2016). Research has shown that models can superficially perform tasks without learning the underlying reasoning process **?**. 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. Agrawal et al. (2016),Zhang et al. (2016), Fukui et al. (2016).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 Goyal et al. (2017)

model learns biases in training and manages to give good results in the testing Selvaraju et al. (2020). 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." Johnson et al. (2017); This learning problem is crucial because it makes it challenging to evaluate the model's improvementsAgrawal et al. (2018).

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" Monroe et al. (2016). 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."

## 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(Mooney (2008)).

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 can be clearly represented in a visual form. Therefore visual recognition is part of the conceptualization process that form the perceptual symbol of an entity.(Barsalou et al. (1999))

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 attributes that form an entity.(Barsalou et al. (1999)) 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.

The full meaning is the perceptual representation and the knowledge about it. The meaning of rotten apple is fully understood when we construct a knowledge of the negative aspects of eating it. The knowledge about health implications, and the attributes such as colors and smell of a rotten apple help us categorizing the rotten apple in the category of rotten food. This attributive characterization can be expressed for example in the way we do prototyping and categorization of entities.(Lakoff & Johnson (2008)).

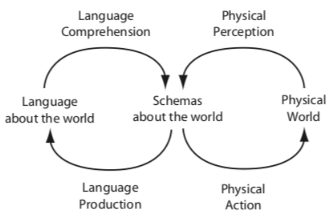



Figure 1: Roy (2005)

.

Interactions in the semantic world has an exchangeable nature. The interaction allows us to form meaning and the formed meaning shape our language and actions. In 1 we see that the outcomes of our interaction in the world includes not only linguistic implications but also affects the actions(Roy (2005)). 'schemes about the world' is the beliefs we make from the interactions. For example, rotten apples have brown colors, and by eating them we established the schema "belief" that rotten apples are bad. The knowledge of "rotten apples are bad" influence our future actions- makes us not eat the apples with attributes of "rotten".

The ability to understand how different attributes form a meaning denotes the notion of reasoning. In order

Meaning is needed to establish a belief and a belief is needed to make an inference. The ability to draw the conclusion that the apple is rotten required

form a meaning from the physical attributed and

Reasoning is possible when we are able to conceptualize the symbol(comprehend the meaning). Comprehending the meaning and grounding it in language, allow us to categorize the apple as rotten food. The ability to draw conclusions from visual information is

logical relation that links the different representations.

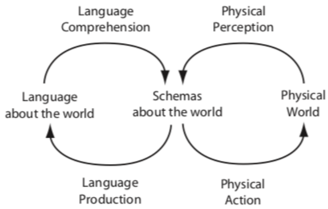



Figure 2: Deb Roy

.

In a more concrete example, a reasoning process would ”require models to perform inference at multiple levels of abstraction” Selvaraju et al. (2020). 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 Mikolov et al. (2013). 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 wordNilsson (2007). To get rich meanings by connecting to the physical world would be to connect language with perceptionMooney (2008).

## 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”Harnad (1990) 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.Lowe (1999).

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.

Larsson (2017) 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 Matuszek et al. (2012) Larsson (2015) we see examples of connecting formal semantics with perception.

Larsson (2017) 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: Larsson (2017) 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.Nikolaus et al. (2019)

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 Larsson (2017) 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.

## image captioning

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

Video Lin et al. (2014)

The two main methods are feature extraction and attention mechanisms Wang et al. (2020).

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.

### 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 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).

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.Vinyals et al. (2015) 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 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. 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

## Feature extraction- statistical language model

Statistical language model, as in Fang et al. (2015), 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)Zhang et al. (2005). 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.

## Attention-mechanisms

In the previous section the discussion on the methods and implementation

## 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 et al. (2017) trains a system in encoder-decoder model on a data set of 2 pairs dialogue with an image content.Skoaj et al. (2011) 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.

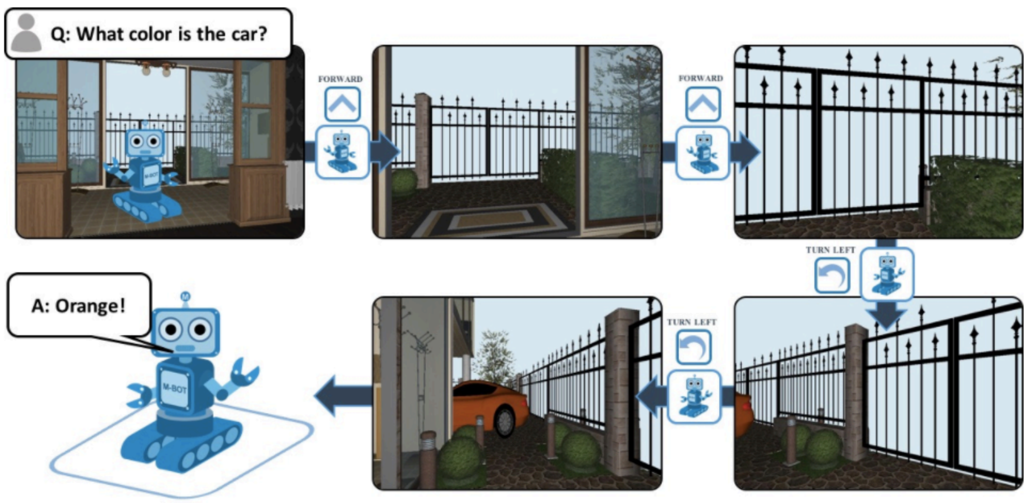
Antol et al. (2015) is the first notable data-set published for Visual Question answering (VQA). The data-set consist of open-ended and free-form questions. The data contains 250,207 images from MS COCO Lin et al. (2015) and other abstract scenes.The question types in the dataset require a range of different 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) 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 Krishna et al. (2016) 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

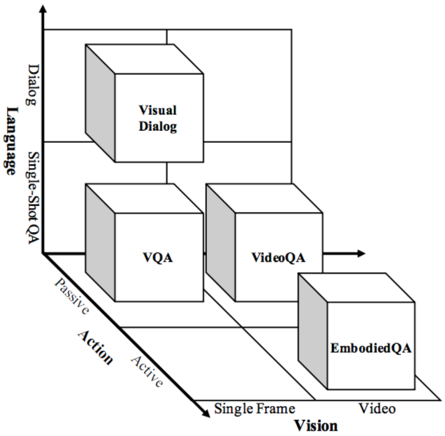
## EQA




.

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.Grisetti et al. (2010) Dissanayake et al. (2001) Zhang et al. (2018)

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).

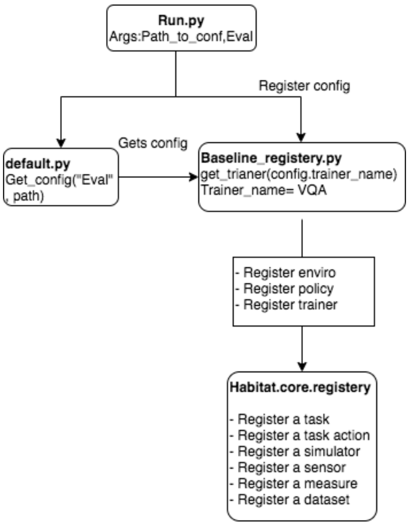
# Methods and materials

## Habitat Simulator and Lab- Overview

We use the Habitat platform as a host for the EQA task. The name 'Habitat' is derived from the notion of learning within and from an environment. Imitating our natural habitat, the Habitat platform 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 an 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. 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.




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 data-set which takes part in lab module.

Embodied question-answering is part of the tasks included within the Habitat platform. The EQA task consist of two sub tasks. The first is navigation, and the second is Visual Question-Answering. The main idea of the task is that 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.

However, 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 and trained in reinforcement learning. Setting up the task in such a connected manner lead to distorted images and inaccurate view point of the object in question if the robots drifts off the path during navigation. ”Noisy or absent views” would confuse the question-answering model.

The current configuration for the EQA task separates the training for navigation and VQA. Imitation learning is used for navigation, and supervised learning for question-answering. the navigation is frozen when once it completes a navigational episode and is evaluated based on its steps following the shortest path to the objective. VQA model is evaluated based on the answers predicted.

’shortest path’ is used in the training sitting of both, the navigation and VQA. ’Shortest path’ is a navigational path consisting of the shortest steps to take to reach a view point of where the object in question is located. With each question-answer in the EQA data-set we find one shortest path. Shortest paths are made by human workers. For navigation the system is trained on following the steps of the shortest path. For VQA the shortest path is used to capture the scene from the viewpoint at the end of the path.

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.

The vision-language modalities are fine-tuned differently for navigation and QA. Navigation and VQA employ the vision and language modalities provided in the platform. However, each one of them use the two modalities with different configurations, such as the number of decoders for vision and the times that the language encoder is used.

## 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.“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).

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:

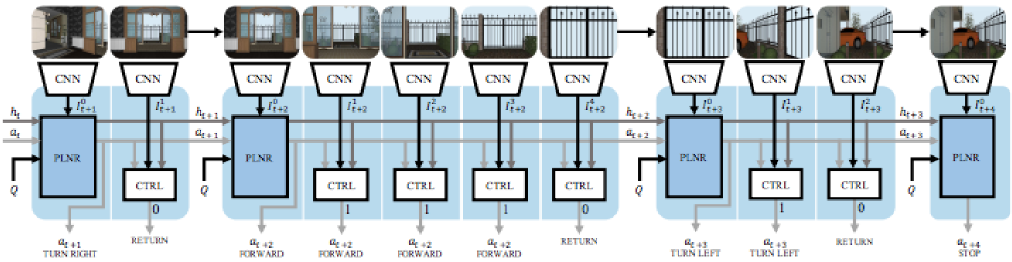
Navigation: ”depth” and ”RGB”. Depth sensor is essential for the agent’s capability to navigate. With depth sensor it could estimate distances and avoid colliding with obstacles.

VQA: The existent baseline VQA model uses the visual information with ”RGB” data only. (No reason mentioned to why the other sensors are not used in the question-answering baseline module).

## 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) "

## 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 :

"$I_t^n$denote 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-state$h_t$ 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.

### Imitation-learning

## VQA

## Data and Data-sets

Our method of generating questions is largely related to the structure of the data-sets in the initial EQA paperDas et al. (2018). Understanding each part of the data-sets and their structure would give an insight into the work flow of generating questions to the task. In this section we elaborate on the source of the data-sets, their content, and our methods in processing the data.

The data-sets 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 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 v1 is also a synthetic dataset generated automatically and can be found at this footnote reference [2]. For generating questions for SUNCG, a code published at this reference[3]. 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’s 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] https://github.com/facebookresearch/habitat-lab
[3] https://github.com/facebookresearch/EmbodiedQA

## Scene Data-set and semantic annotations

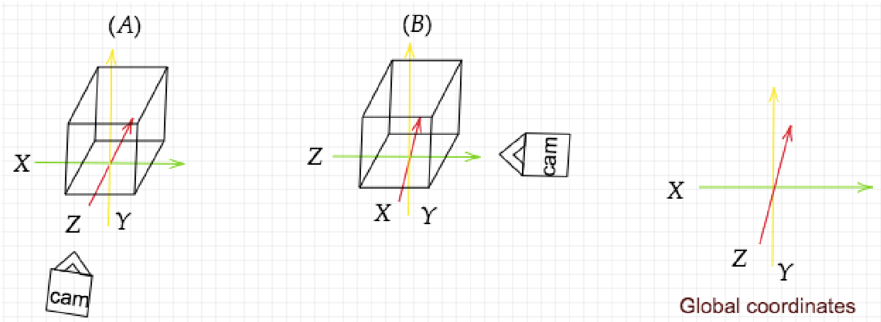
(restructuring is required– more precision) (examples to rephrase– why do we need the location in global coordinates and why the camera views are also important)

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,obj2:..),room2:(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 Chang et al. (2017)(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 , 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:

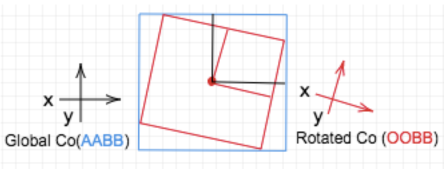
**1. World coordinates**(global): World coordinates(global): The coordinate system that starts at the center of the world; a house in our example. The center of an object in this coordinate system, is then decided by its distance to the center of the world.

**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.




**Processing geometric data- Methods** 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.

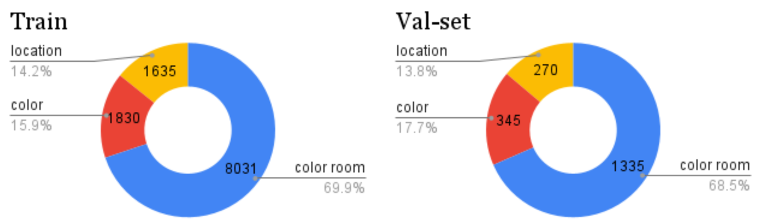
True that the OOBB corners are better representatives of the objects corners, however, locating the oobb corners in the world map is not simply done by measuring how its radius stretches from the center in the direction of the global axis as in AABB. The radius of the OOBB is given along its local view 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.

## 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 et al. (2018) 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 Wijmans et al. (2019).

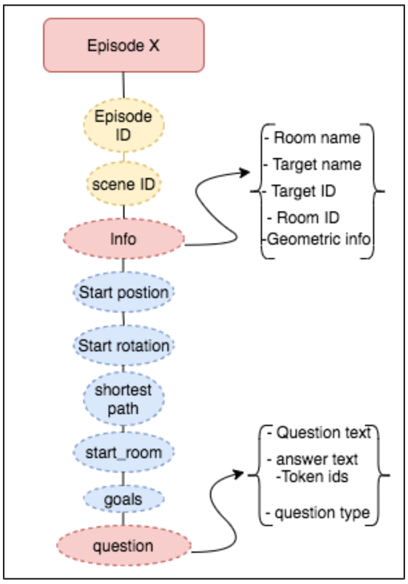



Figure 3:

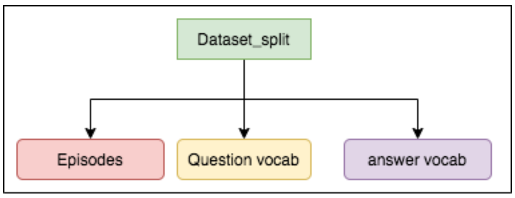



Figure 4:

**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.

## Bias and answer distribution

### Evaluation

**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

# Task 1- Question Generation

## 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 generator. This chapter describes the projects construct and the usage of each part of it.

## 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

The experiment include used two different ways for 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

### **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-

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

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).

## Class 1 Method 2 - Annotations from Habitat's sensors

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 [4]. The configurations include the settings such as the scene, the height and width of the sensors, and the types of sensors to include. color sensor, semantic sensor and depth sensors are used.

Once we simulate the environment, the sensors output the annotations of a house(scene) as an object. We iterate through the object to obtain information about the levels, rooms, and the objects in the rooms. We freeze the simulator once the annotations' object of one environment is outputted, then repeat the process for the other environments.

The data about the an environment's annotations include a calculated geometric information. The sensors in the simulator gives an advantage by calculating the sizes of both the AABB and the OOBB. We use the sizes of the OOBB and AABB, to further obtain more specific information about the object's boxes.

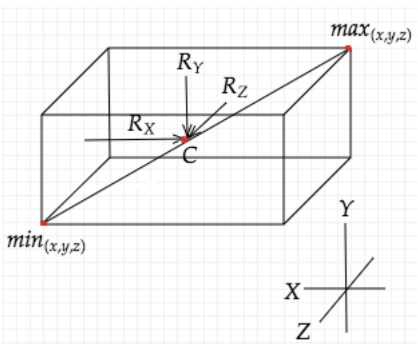



Figure 5: Min and Max of an Axis Orients bounding box

[4]https://aihabitat.org/docs/habitat-lab/habitat-sim-demo.htmlscene-semantic-annotations

We make calculations from the data we extract in order to obtain other necessary info for generating question. Important to mention, the geometric points extracted by habitat's simulator, such as centers of the boxes and the sizes of the boxes' sizes, are represented in 3D data where the x is the length, y is the width, and z is the height.

The first calculation is finding the 'min' and 'max' of a bounding box given an object's center and length of its sides on(x,y,z). In figure 5 we see visual illustration 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 point 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. The given size of the box is from the min point to the max point with a 3d value (x,y,z). To get each of the points we first get the half extent of the size such as: $Half\ extent\ = (x, y, z)/2$ . half extent or (radius) is the point from the center to either the min or max.

The 'min' is the point stretched by the length of the half extent in the negative direction, and 'max' is the stretch to the positive as the following:
$Min\, point\ =\ C\ -\ \vec{H}\ \ = (x\_1 + x\_2, y\_1 + y\_2, z\_1 + z\_2)$
$Max\, point\ =\ C\ +\ \vec{H}\ \ = (x\_1 - x\_2, y\_1 - y\_2, z\_1 - z\_2)$

The min and max help us in define spatial relations among objects. How we use them is explained in the coming sections.

We use the OOBB sizes for calculating the sizes of the objects. We consider the size as the volume of the box which is the length multiplied with the height and width. In our case the length is the x value, width is the y value and z is the height, then the calculated volume of a box is $X\ x\ Y\ x\ Z$.

We structure the annotations and save them in a file. The structure of the data consist of a dictionary storing the data in a hierarchical way. At the top part is the house id, then rooms in the house, then the objects in the house. Each object stored by id, contains the min and max value of its box, size of its aabb, its name, room name and id where its located, and the level id where the room is located. Storing the annotations this way allows to access all the objects in a room through the scene id and room id.

We store the calculated volume of each object in all the houses and store it in a second file. The volumes of objects are stored by their object category. In the volumes file, we find the volumes of all the objects in all of the houses stored in a dictionary, each key represents a category such as 'sofa' with values of the volumes of this object type. The point here is to obtain data on the sizes of each object type. We use this information for finding ground truth answers about for the size questions.

### 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 6 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 .

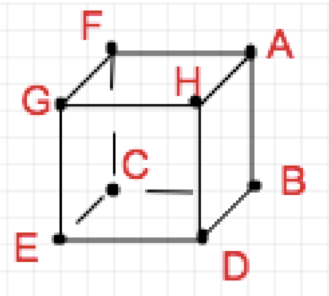



Figure 6:

We 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 6, 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. Each of the spatial relations are determined given a geometric criteria of distances and positions between objects' corners along the three axis (x,y,z).

Two general operations are used in each of the relations. The first one is calculating the Euclidean distance between two corner points; denoted as the distance between p and q in this formula: $d\left(p,q\right) = \sqrt{\sum_{i=1}^{n}\left(q_i - p_i\right)^2}$

The corners, depending on the type of relation we want to extract, can be represented as points in 1d, 2d, or 3d. 1D is when we pick points from one of the axis only. 2D is a point on two axis, and 3D is a point on three axis as the corner examples illustrated earlier. We describe this in more detail in the sections below. The second operation is calculating if one of side of a box on a certain axis is contained within the other.

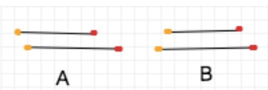



Figure 7:

The calculation if one side is contained within the other relies on defined criteria. The calculation is done in a specific function and can take as input corner points on one axis or more. In the drawing 7 'A' represents two lines on the 'X' axis where the top part is not contained within the other, an in 'B' the top is contained within the lower line. In this example we determine the 'contain' relation by taking the 'min' represented by the the orange dot and the 'max' doted in red.

If the min of the upper line is greater than the min of the lower line and the max of the upper line is less than the max of the lower line then the upper line is contained within the lower. Hence the 'min' of the upper line is greater than the 'min' of the lower line because it's more to the right in the positive direction of the 'x' axis. If the previous condition is not satisfied, then the lines are otherwise not contained.

The operations above are done over different axis for every relation. Below we specify how each of the 'on', 'next to', and 'close' relation is determined between the objects.

**On** First step is choosing pairs of objects that are closest to each other vertically(on the Z axis). The distance should be less than a 0.5 millimeter thresh hold. The Euclidean distance here is calculated between the 1D points on the Z axis only. Two corner values from each box and any corner match the distance required are picked out.

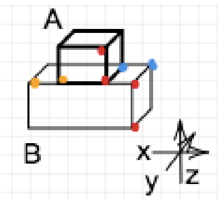



Figure 8:

The second step consist of a group of conditions that the pair of objects need to meet in order to be considered on each other. The first condition is that the vertical sides, the line from Zmin to Zmax, of the boxes are not contained within each other. The Zmin-Zmax lines of every object box are the lines between the two red dots in box A ans B in the illustration 8. Otherwise if the lines on the Z axis are contained it would mean one object is inside the other.

The second condition is that the horizontal line of one of the boxes is contained within each other.The horizontal lines in the illustration are from the Orange to the red points in each box. The lines on the y axis from the red to the blue points should also be contained.

The final step is deciding which object is on the top of the other. The pair of objects are passed to a function that see which Zmin-Zmax has greater value. The object on the top should be in the upward positive direction.

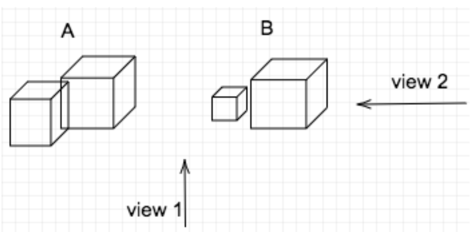



Figure 9:

**next to** A pair of objects next to each other have to have a distance not greater than 0.1 meters on the X and 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 pairs should not have contained sides in neither the x nor the y axis. In this condition a pair of objects next to each other would like illustration A in 9. This is 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 view point. From view 1 the pair(B) seem next to each other but from view 2 they would not. In pair(B) from view 2, one object would be behind the other and likely hidden. So if view 1 is the global view and we pair the objects as in (B), seen as next to in view 1, the robot might enter the scene from view 2 and it would be wrong to refer to the pair as next to each other. However, if the ’next to’ pairs are assigned as in illustration (A), the pair would be still visible in whichever view, and positioned proper enough to be referred to as next to each other.

Finally the pair must have their lines contained 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 who has any of their 3D corners close to each other within a max distance of 0.2 meter. No other conditions required for the pairing of objects close to each other. It can be a close object on, above, below or next to.

## Second Module - question generation

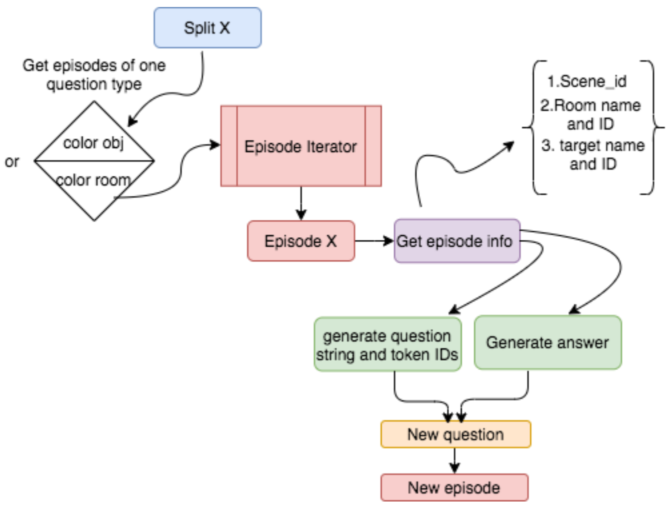



Figure 10: Split generator

Our question generator generates questions of two types, size and spatial. 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 the code, one separate time for each type.

The split generator is the core component in the code. It takes a split either train/val and a question-type as arguments. The functionality of the split generator is to turn an EQA v1 split of episodes into a new split of new episodes. In figure 10 we see an illustration of the workflow of the split generator. The five general steps is filtering(uncolored rotated square), iterator(red rectangle), episode parser (pink rectangle), QA generator (the green rectangles), and episode wrapper (the bottom yellow rectangle- inputs QA and outputs episode )

The filter returns a set of question-episodes of one type only. The returned set of questions of a type is dependent on the question type given to generate. For example, if the input is to generate questions of ’size-room’, ’how big is the sofa in the room’, we take only the questions of ”color-room” type.

The filtered set is passed to an iterator. Each iteration passes one episode from EQA-v1 to a parser. The parser function in the iterator extract information, such as the object name and id, scene ID and room ID, from the EQA-V1 episode.

The parsed information is passed to a QA generator. The QA generator is better described as a group functions of the split-generator that are conditioned differently dependent on the question type. The answer generation function is, however, a different function for each question type.

The general idea for generating QA of any type relies on two straightforward steps. First, generating 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. Second step generating a question string and token ID’s.

The final step consists of inserting the new question with the corresponding geometric information, and structuring them into an executable function. We call a QA sample an episode when the section of the episode seen in figure  are filled with the new QA and the other the corresponding information.

Our question generation can be described as generating one question for every “shortest path” there is. The idea of transforming an episode from EQA into a new one is based on using the starting position and the ’shortest path’ found in them. Having more questions for each shortest path is equivalent to having multiple questions about the same scene.

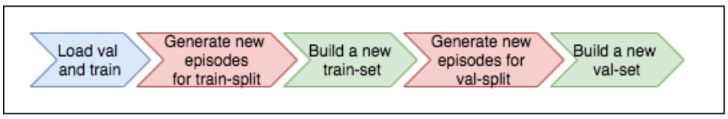



Figure 11: Split generator (The top part of the code)

The top most part of the code (the data-set generator), illustrated in figure11 , passes the train and val to the split generator at different time-stamps. 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 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.

Once a split of episodes is generated it’s passed to loader function. The loader function inserts the answer and question vocab to finalize the data-set in the form seen in figure 2.2

In the coming two sub-section we describe in detail how the QA generators for the size and spatial questions work.

### 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 from the original EQA-v1 episode, as described in the previous section.

### 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 the size of its box and returns an answer about its size. The function calculates the volume of the target object in a similar way as the rest of the sizes of the objects. Volume of the OOBB = 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.

The relative size is determined by its deviation from the standard of its type. As mentioned earlier the sizes of all objects are stored by type in a file. We pick the volumes of the object's type and calculate the mean size and the standard deviation of all the the sizes from the mean. The standard deviation denoted below:

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

The answer is 'small' if the objects' size is smaller the mean 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.

We control the answers' distribution. We observe that a majority of objects have a medium 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 type and the split (train or val), and is based on our observation of the answer distribution in the splits. Note that we refer to 'question type' in this example if either the question to be generated is size question with string 'room' such as "color-room" or without.

**question-string** The question string generator takes a question type, and an object as an argument and returns a complete question string.

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?'

**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 an room id to the 'relation' extractor to obtain pairs of objects, within a room, with a spatial relation between them. The relation extractor returns three types of relational : next, on, or close, if existent within a room. Else it returns a category with empty values.

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 . 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.

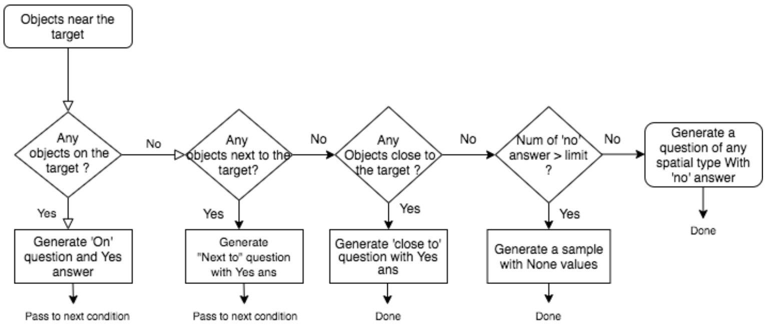



Figure 12: Decision tree for generating different types of spatial questions

A QA with positive spatial answer has a 'yes' answer, and 'no' if a relation is non existent. The decision tree as seen 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 'shortest path' in an EQA 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. These arguments are seen in the input section in the illustration in fig 13. Such as potential object type, spatial question type, and all object in a room

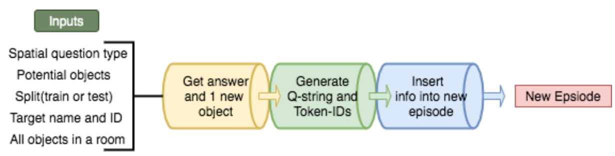



Figure 13: Structure of spatial questions generator

All the inputs seen in 13 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's 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.

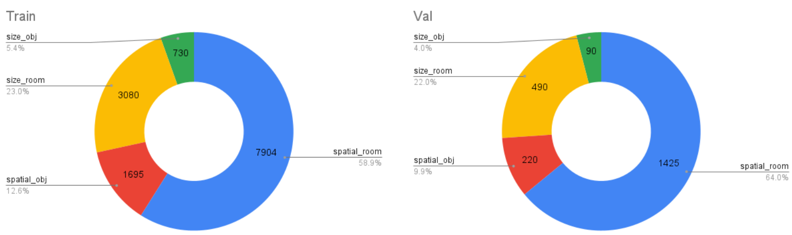
The last part in blue is conditioned by the type of answer, if it's 'yes' or 'no'. If the answer is yes, geometric information of the target object's pair is passed to it to insert it in the episode. If the answer is 'no', no additional information is added to the episode beside the information of the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

**Question strings** To generate a spatial question string, the string generator takes as an argument the question type such as "spatial_room" and and spatial relation type such as "on", "next" or "close". The string 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?'

## Results

### Total number of generated questions




We generate a total of 13 409 question for train and 2 335 questions for validation. In figure **??** 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?".

### Answers distribution

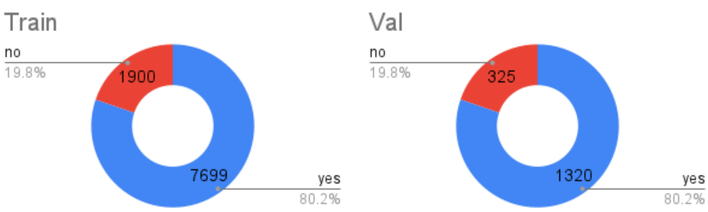



Figure 14:

**Answers distribution of spatial questions** The majority of answers for the spatial questions are positive ”yes” as seen in figure 15. This imbalanced distribution is an intended outcome. The motivation behind this intention is based on an idea, formulated by Regier (1996), of learning of positive samples only. The argument behind this approach to learning is inspired from a cognitive theory of human’s first acquisition of language. The theory is based on the premise that humans tend to learn spatial relations from positive evidence instead of non existent instances.

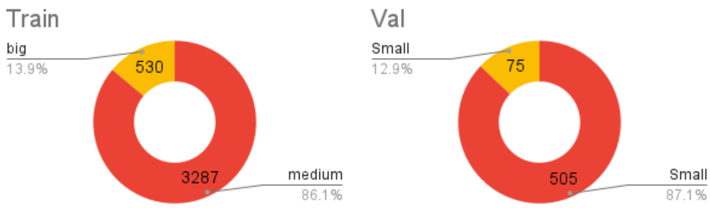



Figure 15:

**Answers distribution of size questions** The outcomes of generating size questions resulted with zero samples of ”small” answer and a majority of ’medium’ answer as seen in figure 15. In the QA generator, we intended to limit the question-answers with ”Medium” answer based on an observation of their dominance. However, limiting the ’medium’ answers more than the presented numbers would have resulted in a very few question-answers of 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.

# Discussion

Our choice to include size and spatial questions is motivated by a ... “spatial language and spatial cognition”(Landau & Jackendoff (1993)). 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-answer generation is very dependent on the ”shortest paths” in EQA-V1 data-set and the objects they lead to.

# Task 2- Question Asking

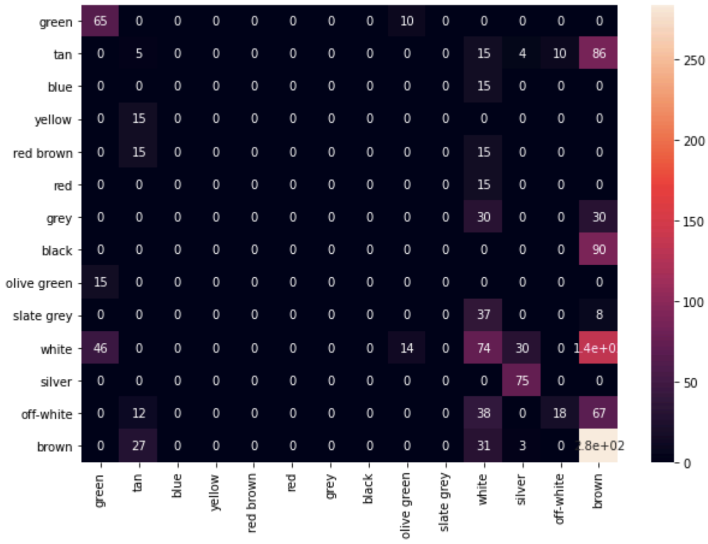



Figure 16: Predictions of color questions before training with new questions

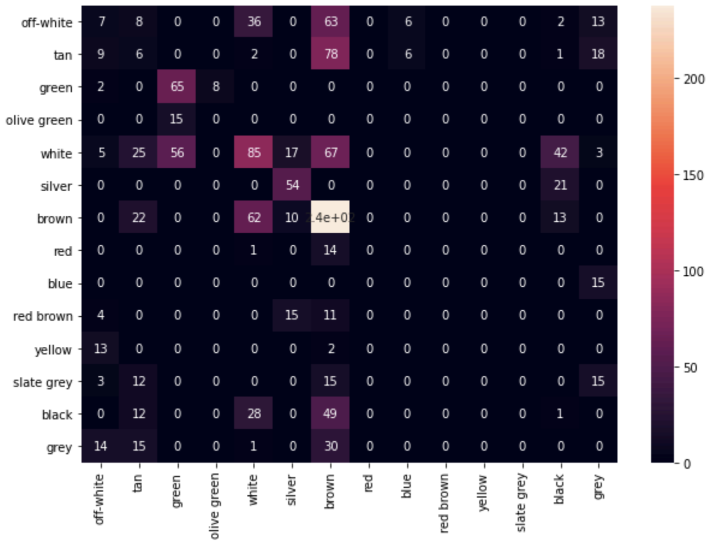



Figure 17: Predictions of color questions After training with new questions

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| no | 0.58 | 0.59 | 0.59 | 317 |
| yes | 0.90 | 0.90 | 0.90 | 1316 |
| accuracy | | | 0.84 | 1633 |
| macro avg | 0.74 | 0.74 | 0.74 | 1633 |
| weighted avg | 0.84 | 0.84 | 0.84 | 1633 |

Figure 18: Scores for spatial questions

Discussion

# 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.

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

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.

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.

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.

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).

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.

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

Landau, B. & Jackendoff, R. (1993). Whence and whither in spatial language and spatial cognition? *Behavioral and brain sciences*, 16(2), 255–265.

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

Larsson, S. (2017). Compositionality for perceptual classification. In *IWCS 2017 — 12th International Conference on Computational Semantics — Short papers*.

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*.

Monroe, W., Goodman, N. D., & Potts, C. (2016). Learning to generate compositional color descriptions. *arXiv preprint arXiv:1606.03821*.

Monroe, W., Hawkins, R. X., Goodman, N. D., & Potts, C. (2017). Colors in context: A pragmatic neural model for grounded language understanding. *Transactions of the Association for Computational Linguistics*, 5, 325–338.

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.

Nikolaus, M., Abdou, M., Lamm, M., Aralikatte, R., & Elliott, D. (2019). Compositional generalization in image captioning. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)* (pp. 87–98). Hong Kong, China: Association for Computational Linguistics.

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

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

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).

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

Skoaj, 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).

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)*.

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.

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

# Appendices