university of groningen / faculty of arts

PREQA: A Photorealistic Dataset for Embodied Question Answering

Frieso Turkstra

Master thesis
Information Science
Frieso Turkstra
S3788946
June 30, 2024

ABSTRACT

A new PhotoRealistic dataset for Embodied Question Answering, or PREQA for short, has been developed. PREQA is a high-quality dataset that is designed to bring the task of Embodied Question Answering (EQA) as close as possible to the real-world. As such, its primary focus is to be used as a test set for pre-trained EQA systems. The dataset consists of 2D images of a real-world robot lab and introduces new features such as camera tilt, viewpoint-based navigation and novel annotations. The dataset contains 1,910 automatically generated questions about three rooms and 90 objects that have been annotated for their class, location, colour, and spatial relationships to other objects. Furthermore, an EQA system is developed and tested on PREQA. This system is used to showcase the benefits of the camera tilt and a new optimal path navigation strategy. More generally, the aim of PREQA is to demonstrate a method to construct EQA datasets from 2D pictures of real-world environments.

CONTENTS

1 INTRODUCTION

An important goal of AI research is developing an embodied system that can perceive its environment, communicate and reason about the environment, and, ultimately, act in that environment. This goal is becoming more relevant as personal assistant and care robots become more prevalent. Embodied Question Answering (EQA) is a task that, if solved, can be seen as one step towards such an intelligent system. The goal of EQA is to build a system that can navigate through an unseen environment in order to answer questions about that environment. To illustrate the challenges of the EQA task, consider the following question: "Is there a brown book in the office?". To answer this question, the agent needs to know what a "book", an "office" and the colour "brown" is. Then, the agent must understand it needs to navigate to the office, explore the entire room, detect whether there is a book, check if its colour is brown, and then come back to the user to respond either positively or negatively. This is a complex task that requires an understanding of language, vision, and navigation as well as commonsense reasoning and long-term memory.

The present study seeks to advance the development of competent EQA systems by introducing PREQA: a new PhotoRealistic dataset for Embodied Question Answering. The dataset contains 96 pictures of a real-world robot lab. These pictures and the objects they depict are manually annotated and these annotations are used to automatically generate questions about the robot lab. PREQA is a high-quality dataset that is designed to be as close as possible to the real-world. As such, its primary focus is to be used as a test set for pre-trained EQA systems. Compared with earlier EQA datasets, the most notable features of this dataset are:

- Photorealism: the images in the dataset have not been taken from a virtual 3D environment but from a real-world robot lab;
- Camera tilt: two new navigational commands to look up and down;
- Viewpoint-based navigation: a method to mimic movement through a 3D environment by connecting images through viewpoints;
- New annotations such as the level of occlusion and object instance identifiers.
- Inclusion of ambiguous questions.

The main contribution of this thesis is the development of a new, high-quality EQA dataset. Additionally, an EQA system is developed to perform extensive experiments with some of the new features of PREQA. First of all, the novel occlusion annotation is used to demonstrate an improved gold standard for navigation in EQA, viz., navigating to the optimal rather than the closest view of the target object. Furthermore, the automatically generated questions are fed into a text-only language model to reveal any exploitable surface-level relationship between the questions and their answers. Lastly, a comparison is made between the difficulty of various question types.

The thesis is structured as follows. First, an overview is given of the EQA task in general, the current state-of-the-art, and earlier EQA datasets. The rest of the thesis can be split up into two parts: the first part concerns the creation of the dataset. This starts from the data collection and data annotation process, continues with the question generation part, and ends with how to evaluate the performance of an EQA system on this dataset. In the second part of the thesis, extensive experiments are conducted with the new features of PREQA. Lastly, limitations and future research directions are discussed.

2 BACKGROUND

2.1 TASK DESCRIPTION

The task of embodied question answering has its roots in regular Question Answering (QA), a long-standing task in the field of natural language processing which dates back to 1959 (Simmons, 1965). The goal of a QA-program is to answer questions in natural language by retrieving the relevant information from some stored data. This is nowadays known as extractive QA in which the answer is directly retrieved from the context. Generative QA, on the other hand, concerns models that generate free text, either still based on a context (open QA) or not (closed QA). QA is still an active research field, as illustrated by the inclusion of the task in modern benchmarks for large language models such as SuperGLUE (Wang et al., 2019). Modern QA systems have also expanded to multiple modalities, leading to a new task called Visual Question Answering (VQA).

A VQA-program takes a multimodal input, consisting of an image and a text-based question, and outputs an answer to the question based on the image. Although the task of open generative VQA was proposed quite recently by Antol et al. (2015), the idea of multimodal QA was already present in Simmons (1965), whose definition of the term *question-answering machine* includes "machines which generate sentences in response to pictures" (p. 53). However, his definition did not include the task which the present study aims to address, namely embodied question answering.

The Embodied Question Answering (EQA) task extends the traditional VQA task by introducing movement, turning away from static 2D images towards a dynamic 3D environment. The task is defined as follows: "an agent is spawned at a random location in a 3D environment and asked a question ('What color is the car?'). In order to answer, the agent must first intelligently navigate to explore the environment, gather necessary visual information through first-person (egocentric) vision, and then answer the question ('orange')" (Das et al., 2018, p. 1). The addition of a third dimension and movement introduces numerous complexities. Namely, in addition to understanding language and vision, an EQA-program must actively perceive its environment, decide where to go based on its perceptions of the environment as well as commonsense reasoning; if the agent is asked about the color of a refrigerator, it is probably best to first find the kitchen. Furthermore, it needs to remember information from previous frames, introducing the need for some long-term memory. A key feature of EQA is the fact that the agent can only use the raw RGB data of the pictures, it has no internal map representation of its environment.

There are numerous tasks that either target a specific component of EQA, extend the task by introducing new elements, or modify the task by replacing certain elements. These tasks, as well as their findings, limitations and opportunities, can roughly be divided into two categories since they either concern the navigation or the visual question answering. These two categories are discussed next, followed by an overview of earlier EQA datasets.

2.2 NAVIGATION

How the robot should navigate depends on the type of questions. For example, Das et al. (2018) only built a system for mono-target questions, i.e., questions that target a single object such as: "What colour is the car in the garage?". For these

type of questions, the navigation boils down to moving directly to the target object, which is similar to the objective of a task called ObjectNav. The goal of ObjectNav is to find an object based on its label, where "finding" either means being within a specified distance of the object or having the object within the current frame of view (Batra et al., 2020). One difference between ObjectNav and the navigation part of EQA is that the former only works with object classes and not object instances. For example, a robot can be asked to navigate to "a chair" but never to "this red chair". In QA, it is common to ask about object instances so PREQA includes both questions that target object classes and questions that target object instances.

Yu et al. (2019) extend the EQA task to multi-target questions such as: "Is the table in the living room taller than the table in the bedroom?". Multi-target questions complicate the navigation which now requires more long-term planning. In addition to this, multi-target questions introduce the need for new abilities such as comparative reasoning in the example above. PREQA includes both mono- and multi-target questions.

The EQA task has also been adapted to include multiple agents that navigate the environment simultaneously and share their knowledge in order to answer the question as efficiently as possible (Tan et al., 2020). However, since PREQA is based on static images of a real-world robot lab, it is not suitable for multi-agent EQA.

Typically, there are four navigational commands in EQA: move forward, rotate left, rotate right, stop. Furthermore, no strafe or backward movements are allowed. The problem is that there are no standard implementations for these commands. Das et al. (2018) started with defining moving forward as 0.25 meters and rotating in increments of 9 degrees. However, Yu et al. (2019), for example, used 30 degree increments to counter the longer navigations that resulted from the inclusion of multi-target questions. PREQA uses the same navigational commands but in a viewpoint-based navigation system, which is explained shortly hereafter. Additionally, two novel navigational commands are introduced by PREQA: tilting the camera up and tilting the camera down.

Qi et al. (2020) introduced a new task called REVERIE: Remote Embodied Visual Referring Expression in Real Indoor Environments. The task involves interpreting a natural language instruction to navigate to a remote object, i.e., an object that is not visible in the initial frame of view and thus requires navigation. Upon arrival, the robot must choose the right object among several distractors and output a bounding box that frames the correct object. Qi et al. (2020) also introduce a novel point-based navigation appraoch for this task. The approach works as follows: in each room that the robot can explore, a few fixed points in space are chosen from which a panoramic picture is taken. The robot can navigate from point to point and receives access to a new panoramic view each time it reaches a new point. This new point-based navigation system allows the use of actual photographs instead of virtually rendered rooms and objects.

2.3 VISUAL QUESTION ANSWERING

Within EQA, there exist various ways to create and ask the questions. Das et al. (2018) execute questions as functions on the environment. For example, the function `environment.select("chair").singleton().query("colour")` returns all the unique red chairs in the environment. By using functions, the distribution of answers can be tightly controlled and any language bias in the questions is avoided.

Language bias in VQA refers to the phenomenon where models exploit the surface relation between the question and the answer without consideration of the visual input (Yuan, 2021). For example, models will typically answer the question "What colour is the banana?" with "yellow", even when the banana on the accompanying image is green. This means the model has not learned the deeper relation

between the question and the colour of the pixels in the image. In general, VQA models attend more to the language input than the visual input (Goyal et al., 2017). However, Ilinykh et al. (2022) show that by adding noise to the images that models do use the image for low-level knowledge about general structures but to a lesser extent for high-level details.

Another approach to counter the language bias is to use image-captioning techniques to convert the image into text (Bi et al., 2023). Since both the question and the image are now text, there can be no bias towards one modality. Furthermore, since multi-modal language models tend to be larger, image captioning allows for smaller models that can achieve the same results with greater efficiency (Salaberria et al., 2023). However, since image captioning always introduces some information loss, Vision-Language Models (VLM) are still preferred (Ren et al., 2024).

The advantage of using language models is two-fold. The first benefit is that both the question and answer are in natural language, providing an intuitive way for users to interact with the system. Secondly, language models can leverage the external world knowledge they learned from their training data. The downside of this is that language models often lack a proper inner representation of the current environment. Therefore, Tan et al. (2023) construct a knowledge graph of the visual input and combine it with external knowledge. This knowledge graph can serve as a memory-storage which is especially useful for multi-turn dialogues since it reduces repetitive navigation. Nevertheless, the current state-of-the-art EQA systems are not using knowledge graphs but transformers with language-vision alignment (Luo et al., 2024).

A middle ground between questions as functions and free-form natural language questions, is the use of template-based questions (Kafle et al., 2017). This has the benefits of still allowing for automatic generation but also using natural language to stay close to the final deployment setting. Moreover, the form of the questions is restricted by the templates which means a certain level of control can still be exerted over the distribution of possible answers.

An important but difficult topic in both VQA and EQA is grounding. In VQA, grounding is connecting the answer to the corresponding image-data on which the answer is, or should be, based (Zhu et al., 2016). A variety of methods to ground an answer in reality have been proposed. In visual language navigation, an answer is grounded in the location of a robot (Anderson et al., 2018). In VQA, the attention weights that process the visual input have been used as a means of grounding (Zhang et al., 2019). For EQA, Das et al. (2018) ground the answer in an action; "kitchen" means navigating to the kitchen. However, it is possible that the robot can answer a question about the office while standing in the hallway. Furthermore, what if the robot is already in the kitchen? Then no action needs to be performed and the answer is not grounded. What matters, then, is that the robot bases its decision on the relevant visual input. Therefore, in Qi et al. (2020), the answer is grounded in the detected bounding box of the target object. However, this method is limited to models that output bounding boxes.

To test the navigation modules and visual question answering models described above, EQA datasets are needed. The following section provides an overview of earlier EQA datasets.

2.4 PREVIOUS DATASETS

EQA datasets are not merely lists of question-answer pairs such as in QA or triples of image-question-answer pairs such as in VQA since an entire environment is needed through which the robot can navigate. Then, questions can be asked about that environment, its rooms and the objects in those rooms. Most environments are created by using virtual 3D rendering software (Duan et al., 2022).

The advantages of using virtual environments are abundant: they are cheaper, easier to create with more control over the environment, faster training times, and so forth. The major drawback, however, is a lack of realism, creating a gap between the training setting and the final deployment setting. There are photorealistic 3D simulators such as Matterport3D (Chang et al., 2017) but this realism comes at the expense of the ease of creation. Wu et al. (2018) have turned the static SUNCG 3D dataset (Song et al., 2017) into an EQA dataset by turning it into a dynamic environment through which the robot can navigate. It contains a total of 45,622 environments, illustrating the ease of creation. Incidentally, this is the virtual simulator used in the original EQA task (Das et al., 2018).

Gibson is a state-of-the-art dataset of 3D spaces that has tried to close the gap between virtual and real-world environments by using 3D scanning techniques (Xia et al., 2018). This yields highly realistic imagery but has faced some challenges as well such as sparse scan locations, difficulty with reconstructing small objects, and reflective surfaces leaving holes in the 3D mesh (Xia et al., 2018).

The last dataset discussed here is RoboTHOR (Deitke et al., 2020). The dataset is one of its kind because every one of its virtual environments has a physical counterpart. In other words, Deitke et al. (2020) have carefully reconstructed all objects and pieces of furniture so that the virtual environment resembles its physical counterpart in reality. Moreover, in an attempt to democratise the research in embodied visual AI, Deitke et al. (2020) allow researchers to test their virtually trained EQA systems in the real-world environments by remotely controlling the robot. Since the virtual environments are also publicly accessible, one's EQA system can be tested both in the physical world and its virtual replica to test the generalisation of virtually learned skills to the real-world. Deitke et al. (2020) find that there is still a significant gap between performance in the virtual and real environment, emphasising the need for more realistic EQA datasets.

To sum up, PREQA incorporates both mono- and multi-target questions that can target both object classes and object instances. For the navigation, the point-based approach by Qi et al. (2020) is modified to fit the navigational commands of the EQA task. Furthermore, these existing commands are extended with tilting the camera up and down. In Chapter 6, some of the challenges found in the literature concerning language bias and grounding in VQA are addressed. By doing all of the above, PREQA utilises the current state of the EQA research field to demonstrate a method that can construct an EQA dataset from 2D images of a real-world environment. The details of this method are detailed in the following chapters.

3 DATA COLLECTION AND ANNOTATION

To create PREQA, image data needs to be collected and annotated. The first step, however, is to outline the structure of an EQA dataset, i.e., a dataset that is designed for the task of embodied question answering. An EQA dataset can be represented as a set of environments. Each environment is a set of connected rooms that are populated with various objects. Furthermore, each environment is associated with a list of questions about the objects in that environment. Formally, then, an EQA dataset $\mathcal{E}$ can be defined as follows:

$$\mathcal{E} = \{E_1, E_2, \ldots, E_n\}$$

Where:

- Each environment E_i is a triple (R_i, O_i, Q_i).
- Each R_i is a non-empty set of rooms $\{R_{i,1}, R_{i,2}, \ldots, R_{i,m}\}$.
- Each O_i is a non-empty set of objects $\{O_{i,1}, O_{i,2}, \ldots, O_{i,k}\}$.
  - Each object $O_{i,j}$ is associated with certain attributes and may participate in predicate relationships with other objects from O_i or rooms from R_i.
- Each Q_i is a non-empty set of questions $\{Q_{i,1}, Q_{i,2}, \ldots, Q_{i,p}\}$.
  - Each question $Q_{i,p}$ is a question about the objects in environment E_i and/or their attributes or relations to other objects.

Depending on the EQA dataset, objects are assigned different attributes and participate in other types of predicate relationships. The specific attributes and predicate relationships of objects in this dataset are:

- $in\_room(R, O)$: a two-place predicate with a room R and an object O such that object O is located in room R.
- $colour(O, Colour)$: a two-place predicate with an object O and a colour $Colour$ such that object O has colour $Colour$.
- $class(O, Class)$: a two-place predicate with an object O and a class $Class$ such that object O belongs to the class $Class$.
- $on(O_i, O_j)$: a two-place predicate with an object O_i and an object O_j such that object O_j is positioned on top of O_i.
- $above(O_i, O_j)$: a two-place predicate with an object O_i and an object O_j such that object O_j is positioned above O_i.
- $below(O_i, O_j)$: a two-place predicate with an object O_i and an object O_j such that object O_j is positioned below O_i.
- $next\_to(O_i, O_j)$: a two-place predicate with an object O_i and an object O_j such that object O_j and O_i are positioned next to each other.

The annotations of these attributes and predicate relationships are discussed in Section 3.2. Before that, it must be mentioned that the dataset presented here differs in several important regards from a typical EQA dataset. The main differences are:

A Photorealistic Environment

The rooms and objects in this dataset are based on a real-world robot lab. As such, the images of the rooms and objects are actual photographs instead of views from a virtual 3D environment. This photorealism brings the task closer to the setting in which an EQA system would actually be deployed. Since this dataset is based on a real-world robot lab, it only contains one environment with three rooms and ninety objects, instead of the more numerous number of rooms in virtually constructed environments. Therefore, the dataset is not suitable for training. The idea, then, is to train an agent in virtual environments since they are faster, cheaper and much easier to create and control. Ideally, the trained agent would be tested in an unseen, non-virtual environment in order to evaluate how well it generalises to the real-world. However, not everyone has a robot and a robot lab at their disposal. By creating this dataset, other researchers can remotely test their EQA systems in our robot lab.

Viewpoint-Based Navigation and Camera Tilt

In a virtual 3D environment, the current frame of view can easily be extracted based on the position of the virtual robot. This enables the robot to have considerable freedom in its movements, such as rotating left or right by only nine degree increments or moving forward by a maximum of 0.25 meters at each prediction step (Das et al., 2018). In order to simulate this, one would need to take thousands of pictures of the robot lab. Furthermore, all these pictures require manual annotation, rendering this method impractical. Instead, a viewpoint-based navigation inspired by Qi et al. (2020) was adopted.

A viewpoint is a fixed location in space to and from which the robot can navigate. The robot can only move to a viewpoint if it is in its current frame of view and the robot is in the correct orientation. For example, if the robot wants to navigate to a viewpoint to the east but the robot is facing north, the robot first has to turn right by ninety degrees and then move forward. Moreover, the same principles still apply with regards to the robot being only allowed to use the raw RGB data from its egocentric vision. This means the robot is not aware of the existence of any viewpoints, their location nor how they are connected.

At each viewpoint, the robot can rotate around its axis in steps of 90 degrees and tilt its camera up and down by 25 degrees. The rotational degrees of freedom are comparatively more restricted but the novel addition of the camera tilt enlarges the action space again by introducing a new axis of movement. The camera tilt is useful to see objects lying on the ground or attached to the upper parts of the wall or ceiling. This is especially true in narrow rooms where it is hard to find an optimal point of view of nearby objects.

All the images, annotations, questions and code to process and generate all the data, can be found on Github[1].

These two key differences impact all components of the dataset creation pipeline. The effects of these changes, adaptations and new additions are discussed in the following sections and chapters — starting with the data collection and the data annotation process.

3.1 DATA COLLECTION

All pictures have been taken in the robot lab with a Temi robot V3. Both the robot lab and the robot used in this study have been made available by the Center of

[1] https://github.com/Frieso-Turkstra/thesis

Language and Cognition at the University of Groningen. The robot lab consists of two rooms connected by a hallway. A map of the robot lab can be found in Figure 1.

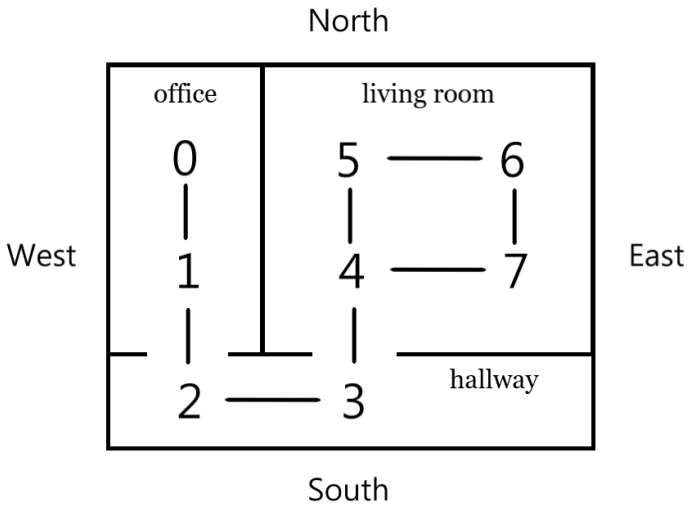


Figure 1: A map of all the viewpoints in the robot lab.

The rooms are decorated with furniture and objects to resemble a living room, an office and a hallway (see Figures 2a, 2b, and 2c). Every picture has been taken with the frontal camera of the robot, a 13 MP RGB camera with a resolution of 1080p at 30 FPS. The camera is positioned in the head of the robot, approximately one meter from the ground, and has a frame of view of 120 degrees.

Returning to Figure 1, it can be seen that each room has been assigned a number of viewpoints. The edges between the viewpoints represent paths that the (virtual) robot can use to move from one viewpoint to the next. In total, there are eight viewpoints. At each viewpoint, pictures are taken in every cardinal direction. This allows the robot to rotate around its axis in steps of 90 degrees. In each direction, three pictures are taken with the camera of the robot tilted at angles -25, 0 and 25 degrees. This method results in twelve pictures per viewpoint, multiplied by eight viewpoints yields 96 unique pictures of the robot lab. All pictures were taken in one session to ensure the same lighting conditions. Lastly, due to the position of the camera at the front of the robot, all images appear mirrored. Hence, each image has been flipped horizontally before starting the annotation process.

3.2 ANNOTATION

After the images have been collected, the annotation process begins. The annotations have been performed with the help of the VGG Image Annotator (VIA) software, developed by the Visual Geometry Group (Dutta and Zisserman, 2019). Furthermore, all annotations have been performed by the author. This means there is no inter-annotator agreement.

The annotation process can be split up into two phases. The first phase concerns the image-level annotations. This phase involves drawing a bounding box around every object in all images and assigning a class label as well as a level of occlusion to each bounding box. The second phase concerns the object-level annotations. This involves annotating the colour of the object, the room it is located in and its spatial relationships to other, nearby objects.

In order to do the object-level annotations, one must first identify all the unique objects. This is not a straightforward task since multiple bounding boxes can depict the same object from different viewpoints, distances or angles. The problem of

(a) Living room.

(b) Office.

(c) Hallway.

Figure 2: Pictures of each room in the robot lab.

resolving which objects are visible on which images is called instance-level recognition. How this problem has been approached here is described in Section 3.3. For now, all object instances are assumed to be known.

3.2.1 Image-Level Annotations

The first step in the annotation process is to draw a bounding box around every object in all images. A bounding box is a rectangular frame that shows the location of an object in an image and is represented by the coordinates of its top-left corner and its width and height. After all the bounding boxes have been drawn, a total of 666 so-called regions have been identified. Then, each region needs to be assigned a class label, which denotes the object in the bounding box, and a level of occlusion, indicating the visibility of that object. Figure 3 shows an example of an annotated image. Next, an enumeration is given of which classes and levels of occlusion there are as well as how and why they are annotated.

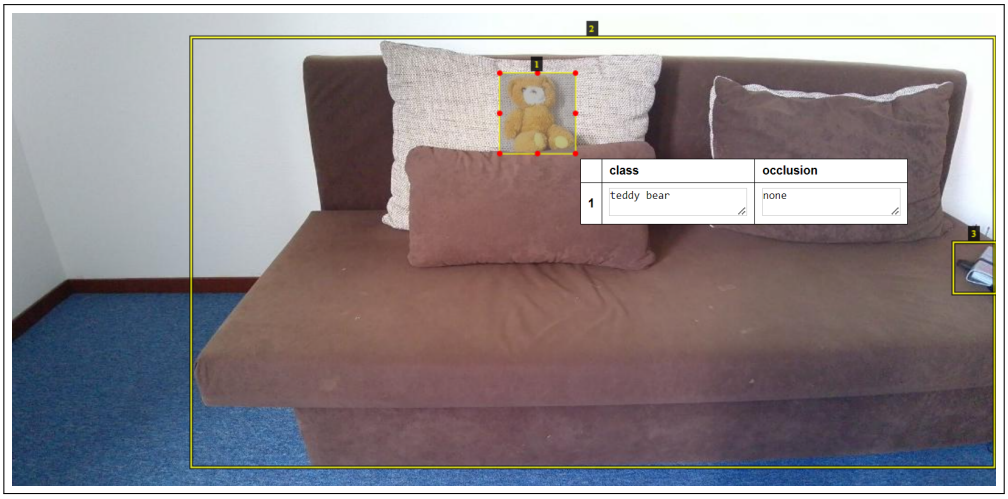


Figure 3: Snapshot of the VIA software with three identified regions: a non-occluded teddy bear, a partially occluded sofa, and a severely occluded bag on the right side.

Object Classes

Technically, the class of an object is an object-level annotation. However, performing the label annotations on the image-level greatly reduces the number of annotations needed to perform the instance-level recognition (see Section 3.3).

The object classes are taken from the original EQA task (Das et al., 2018) who, in turn, adapted it from the SUNCG dataset (Song et al., 2017). However, their classes have very little overlap with the objects in the robot lab as only 8/50 types of objects are present in the robot lab. Therefore, the classes were extended with the objects from the robot lab. The absent objects are still included in the list of classes because otherwise, the answer to questions such as "Is there a(n) [object] in the house?" would always be positive regardless of the object.

Following Hürlimann and Bos (2016), every class is mapped to a WordNet synset (Miller, 1995). This mapping clarifies the exact semantics of the class as well as enriches the otherwise abstract coordinates of an object's bounding box with both linguistic information and world knowledge about the object. As a consequence, some of the original EQA classes were merged. For example, the classes "xbox" and "playstation" have the same WordNet synset, viz., `console.n.02`. Hence, these classes were collapsed into one.

Furthermore, following Das et al. (2018), uncommon objects were excluded from the dataset (e.g., a key box) and semantically similar categories were sometimes

merged as well even though they did have distinct WordNet synsets (e.g., 'desk chair' and 'lounge chair' both became 'chair'). Unlike Das et al. (2018), tiny objects (e.g., a Rubik's cube) and transparent objects (e.g., a vase) are included in the dataset. Tiny and transparent objects are notoriously difficult for object detection models but have been included as an additional challenge. Both can be filtered out, as well as any of the other classes or attributes, depending on the research task at hand.

In total, this leaves us with 71 unique object classes, 31 of which are present in the robot lab. A list of all classes, their WordNet synsets, and their type-token counts — i.e., how many objects belong to a specific class (type count) and how often those objects appear on the images (token count) — can be found in Appendix A.

Levels of Occlusion

After each bounding box has been assigned a label, it is also assigned a level of occlusion. Occlusion refers to the (partial) obstruction of an object's visibility. This can be due to another object being in front of it or because part of the object falls outside the boundaries of the image. The occlusion of an object depends on the position and angle from which the object is viewed and as such, is an image-level annotation.

The levels of occlusion considered here are: non-occluded, partially occluded, severely occluded. Non-occluded means the object is fully visible. Partially occluded means a part of the object is hidden but the object is still identifiable without further context. Severely occluded means the object is hidden to such an extent that identifying it is hard or even impossible without further context. Figure 3 shows three objects, one at each level of occlusion. In regular VQA, these objects would not be annotated. EQA differs in this respect as further context is available by any combination of moving, rotating or looking up and down. Annotating the severely occluded object then, may hint the robot that there is an object at that location which can be identified if the robot assumes a more favourable viewing point. Again, if the dataset is used for traditional VQA, severely occluded objects can be filtered out.

3.2.2 Object-Level Annotations

Following the attributes and predicate relationships defined in the beginning of this chapter, each object is annotated for its class, colour, location and spatial relationships to other, nearby objects. In addition to this, each class label is mapped to a WordNet synset and each object has a domain identifier which links all the images on which the object occurs. An illustration of an object with its annotations is given in Figure 4.

Figure 4: An image of a brown bag in the office. A ball lies on the ground below the bag (not visible).

.

Figure 5: Annotations for the bag in Figure 4.

Domain identifier:	5
Class:	bag
WordNet synset:	`bag.n.04`
Location:	office
Colour:	brown
On:	[]
Above:	[]
Below:	[8]
Next to:	[]

The domain identifier is the result from the instance-level recognition described in Section 3.3. The class and WordNet synset attributes have already been discussed in Section 3.2.1. The remaining annotations — location, colour and spatial relationships — are discussed separately below.

Location

The location attribute refers to the room an object is located in. The possible locations are: gym, patio, lobby, office, garage, kitchen, dining room, living room, bedroom, bathroom, elevator, balcony, and hallway (see Appendix B). These rooms have been taken from Das et al. (2018) with one extra room added to the list, viz., the hallway. Recall that the robot lab only contains an office, a living room and a hallway. Although technically possible, no object in the dataset occupies two rooms.

Colour

The colour attribute describes the main colour of the object. Since most objects contain more than one colour, only the dominant colour is annotated. The dominance of a colour is expressed as the percentage of surface area covered in that colour. For example, one of the desks in the robot lab has a black base and a wooden top. Since the wooden top is the main part of the desk, the annotated colour is wooden. If there is no obvious dominant colour, an object can be annotated with multiple colours. For example, a Rubik's cube has by design six colours with equal surface area. Hence, each Rubik's cube has been annotated with six colours.

The possible colour values are red, green, blue, yellow, orange, pink, purple, brown, beige, black, white, gray, gold, silver, transparent and wooden. All colour values and their corresponding hexadecimal values can be found in Appendix C. The last two values, transparent and wooden, do not have a hexadecimal value because they are technically not colours but textures. Nevertheless, they have been included here due to the prevalence of wooden and transparent objects in day-to-day environments and due to the difficulty of finding the most appropriate colour otherwise.

Spatial Relationships

Some of the objects participate in spatial relationships with each other. The following spatial relationships have been annotated for: on, above, below and next to. In general, the requirements for two objects to be in a spatial relationship with each other include:

1. The objects are in close proximity of each other;
2. There are no other objects between the two objects;
3. The spatial relationship is non-trivial.

The first requirement is intuitive; two objects do not stand next to each other if they are four meters apart. The second requirement prevents two objects from being in a spatial relationship by transitivity. Consider the following example: an apple, book and cup stand in a row on a desk. The apple is positioned next to the book and the book is positioned next to the cup. By transitivity, then, the apple is also positioned next to the cup. Technically, this may be considered true. However, in the context of question-answering, it is more natural that one would ask what colour is the cup next to the book rather than what colour is the cup next to the apple (which is next to the book that is next to the cup).

Although the "next to"-relationship is not considered transitive, it is considered symmetric. This means that if object A is next to object B, then object B is also next to object A. The above-below relationships, on the other hand, are not symmetric.

It is true that if object A is above object B, then object B is below object A. However, the converse is not necessarily true. If object A is below object B, then it is possible that object B is not annotated to be above object A. For example, a cup is located on a table which means the table is below the cup. The cup, then, is not annotated to be above the table but *on* the table. This is done to conform to the pragmatics of the English language and thus, to the intuition of the user.

Lastly, the third requirement precludes trivial spatial relationships such as every object being below a ceiling lamp which, although technically true, is not valuable information to annotate.

3.3 INSTANCE-LEVEL RECOGNITION

Up until now, it was assumed that all object instances were known. In other words, for every pair of the 666 bounding boxes, it was assumed to be known whether the two objects framed by the two bounding boxes depict the same object instance or not. The task of identifying all object instances is called Instance-Level Recognition (Kalantidis et al., 2011; Liu et al., 2016; Del Chiaro et al., 2019). An illustration of the task is given in Figures 6a, 6b, and 6c.

There are two main reasons why it is useful to know whether two or more objects from different pictures are identical or not. For example, imagine asking a robot the following question: "How many chairs are in this room?" In order to answer this question correctly, the robot needs to know whether it is seeing a new chair or one that it has already seen but from a different angle. The second reason is practical and concerns the annotation process; annotating for example the colour of all 666 object regions is time-consuming and includes a lot of repetitive and redundant work. If a particular picture of a table is annotated to be white, and it is known which other pictures contain the same object, then the annotation for those other pictures can be inferred automatically. This also guarantees consistency of attributes between different occurrences of the same object instance.

A naive approach towards annotating object instances would be comparing every object in each picture to every other object in all pictures and annotating whether they are identical or not. For 666 objects, this results in 221,445 comparisons. In general, the time complexity of this method is $\mathcal{O}(n^2)$, where n is the number of objects. The number of pair-wise comparisons, then, is given by:

$$\frac{n^2 - n}{2} \tag{1}$$

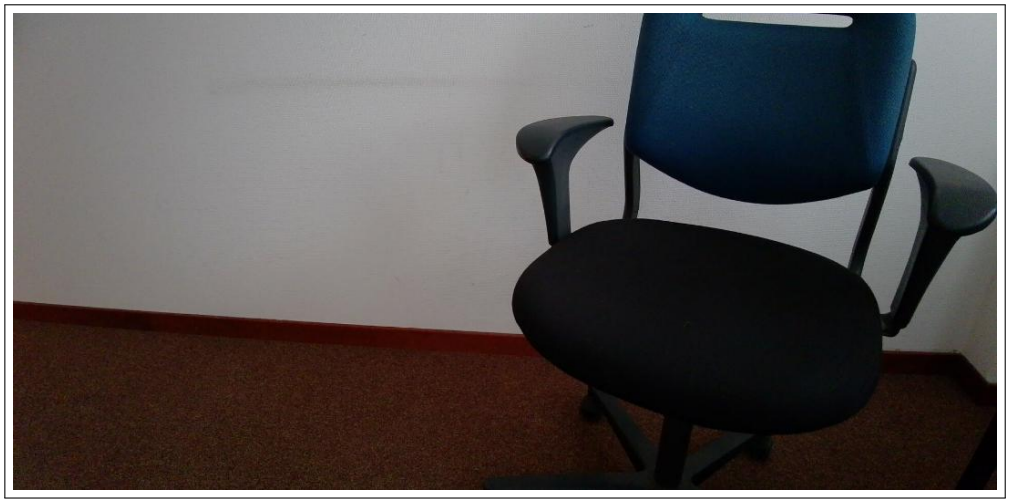
A little less naive approach is a class-aware comparator, which only compares objects that belong to the same class. After all, one should not compare apples to oranges. This does assume the labels are already annotated beforehand, which is why the labels were already annotated at the image-level as discussed in Section 3.2.1. Using the class-aware comparator drastically reduces the number of necessary comparisons from 221,445 down to 10,806. The extent of this reduction depends on the number of unique objects and the number of pictures per unique object. Given the set of objects O, where the i^{th} object occurs n_i times, the number of pair-wise comparisons is given by:

$$\sum_{i \in O} \left(\frac{n_i^2 - n_i}{2} \right) \tag{2}$$

Let m be the number of unique objects and assume each object occurs equally frequent as any other object. Then, the time complexity of the formula above is $\mathcal{O}(\frac{n^2}{m})$. Consequently, the class-aware comparator is certainly an improvement over the naive approach but performing nearly 11,000 comparisons still requires a lot of

(a) One image of the chair.

(b) Another image of the same chair.



(c) An image of another chair.

Figure 6: Illustration of instance-level recognition. The first two pictures depict the same chair. The chair on the third picture, although it looks similar, is a different chair.

valuable human time and effort. Furthermore, this issue will be even more pressing for larger datasets. Therefore, the Union-Find algorithm, also known as the Disjoint-Set Union algorithm, is used to further reduce the number of necessary comparisons.

The Union-Find algorithm was first described by Galler and Fisher (1964). It employs a data structure that stores disjoint sets and has two main operations which are, as the name suggests, *union* and *find*. The union operation merges two disjoint sets into one. The find operation returns the representative of a set. To apply this algorithm to the current use case of annotating object instances, each bounding box is initialised as the representative member of its own disjoint set. Each disjoint set is represented by a tree graph and the root of the tree is the representative member of the set. The annotator is shown two images with one bounding box drawn on each picture and has to mark the objects as identical or not. If they are not identical, no operation is performed and the next pair of images is shown. If the two objects are identical, their sets are replaced by their union.

Suppose object A becomes the root of the new set and object B becomes the child of object A. Furthermore, the next pair of images compares object B to object C. This comparison can now safely be skipped because object C will be compared (or is already compared) to object A, which is identical to object B. More generally speaking, comparisons are only necessary between the representative members of each set. At the end of this process, the number of disjoint sets equals the number of unique objects. The number of members in each set equals the number of pictures on which each object occurs.

By incorporating path compression into the find operation and ensuring the smaller tree is unioned into the larger tree to keep the data structure as flat and by extension as efficient as possible, the amortized time complexity of the find and union operations becomes $\mathcal{O}(\alpha(n))$, where α is the inverse Ackermann function (Tarjan, 1975). This means both operations perform in constant-time for all practical applications. Regardless, in the worst-case scenario when all objects are different, the number of pair-wise comparisons is still n^2. However, in the best-case scenario, when all objects are identical, only $n - 1$ pair-wise comparisons are now necessary. Thus, the precise benefit of using the Union-Find algorithm still depends on the distribution of one's dataset; more images of the same object will result in fewer comparisons.

For the dataset presented here, the Union-Find algorithm could reduce the number of annotations to 1,750 – a reduction of almost 84% in comparison to the class-aware comparator. In addition to this, there are still some more ad-hoc optimizations left that were not implemented here but that might be worthwhile when annotating larger datasets. For example, two pictures taken from the same viewpoint but in opposite directions do not need to be compared because the camera's field of view is less than 180 degrees. This can be extended to neighbouring viewpoints as well; if viewpoint A is connected to viewpoint B in the east-west axis, then any north-facing picture taken from viewpoint A does not need to be compared to any south-facing picture taken from viewpoint B. Lastly, comparisons between viewpoints that are in separate rooms and have no common frame of view can be skipped altogether. In our robot lab, however, it is possible from every viewing point to see the hallway so the optimization was not applicable to this dataset but it might be in environments with more rooms and/or different layouts.

After successfully resolving all object instances using the method outlined above, it turns out there are 90 unique objects in the robot lab. With this information, the rooms and objects part of the environment is complete. Accordingly, attention is now shifted towards the questions part of the environment.

4 QUESTION GENERATION

The environment built in the previous chapter consists of 96 pictures of 3 rooms with 90 unique objects. The questions are automatically generated from this environment using pre-defined question templates. Because the environment is not a 3D virtual environment but a collection of images of a real-world environment, a new, custom question generation engine has been developed. First, the question types are enumerated and the template-based question generation is explained. Afterwards, the dataset is downsampled to make it more balanced.

4.1 QUESTION TYPES AND TEMPLATES

There are seven basic question types and each question type is associated with a template that has a number of slots. Every question has at least one required slot that is filled in with an object from the environment. Additionally, there are optional slots that can be filled in with the location or colour of the required object. Note that the location slot, if it is not filled in with a specific room, defaults to "house". By filling in the optional slots, *variants* of the basic question types are created. In total, there are 24 question variants. A list with all question types and the corresponding templates can be found in Table 1.

Table 1: A list of all question types and the corresponding templates. Curly braces denote required slots and square brackets denote optional slots.

Question Type	**Template**
Existence	Is there a(n) [colour] {object} in the [location]?
Count	How many [colour] {entity} are there in the [location]?
Location	What room is the [colour] {object} located in?
Colour	What colour is the {object} [location]?
Spatial	What is {preposition} the [colour] {object} [in the [location]]?
Conjunction	Is there a(n) [colour_1] {object_1} and a(n) [colour_2] {object_2} in the [location]?
Disjunction	Is there a(n) [colour_1] {object_1} or a(n) [colour_2] {object_2} in the [location]?

Each question type is designed to target specific abilities. The existence question tests object detection whereas the location and colour questions target scene recognition and object attribute recognition, respectively. The spatial question evaluates spatial reasoning and the count question type requires long-term memory and instance-level recognition. Lastly, the conjunction and disjunction questions can be interpreted as simply asking two existence questions. The focus of these logical questions, however, is understanding how to evaluate the truth value of a conjunction or disjunction. As such, these questions target logical reasoning and are considered their own basic question type.

The existence, count, conjunction and disjunction questions do not require the object to actually exist in the environment. This means the answer can be "no" or zero. The other question types, however, do assume the object to exist since a user is unlikely to ask about the colour of a chair if they do not already know that there is a chair in the first place. In general, any question about the attribute of an object assumes the existence of that object. Likewise, any question that specifies a location assumes that that location is present and accessible in the current environment.

The location, colour, and spatial questions, on the other hand, can be ambiguous. Ambiguous questions have different answers depending on which object is targeted, which cannot be inferred from the question alone. Consider the question: "What colour is the chair in the office?". Suppose that there are two chairs in the office, one is blue and the other is black and green. The answer, then, depends on which chair the robot detects first. Such ambiguous questions are intentionally kept in since ambiguity is an inherent characteristic of natural language and it is vital that the robot learns how to deal with it. Depending on the research task, the robot has to provide either one of the correct labels, all of the correct labels or make a clarification request to resolve the ambiguity.

4.2 DOWNSAMPLING

The generation of the conjunction and disjunction questions — and to a lesser extent, the count and existence questions — lead to a combinatorial explosion. This is because questions are generated for every pair of objects, for every pair of colours, for every room. To mitigate this, the combinations of objects are limited by their semantic similarity. This means a question is only generated for two objects if the WordNet synsets of these two objects have a common hypernym that is no more than two levels higher than the original WordNet synsets. This preventive measure alone is not nearly enough to contain the combinatorial explosion. Therefore, in order to make the dataset more balanced, the dataset is downsampled.

The question generation process outlined above yields a highly imbalanced dataset with nearly 94% of questions being either conjunction or disjunction questions. Furthermore, most of the object and colour combinations mentioned in the logical questions do not exist in the robot lab. Consequently, the answer distribution is also highly uneven with the answer almost always being either zero or "no".

To address this imbalance, the questions are downsampled based on the answer distribution. The downsampling is done per question type variant to ensure every question type variant is still present in the dataset. The count questions are downsampled until the answer zero is as frequent as all non-zero answers. The existence, conjunction and disjunction questions are downsampled until the answer "no" is as frequent as the answer "yes". The minority class for the disjunction questions is still comparatively large with over 3,000 positive answers. Thus, the disjunction questions are further downsampled until there are as many disjunction as conjunction questions while retaining the balance between positive and negative answers.

The proportion of question types, both before and after downsampling, is shown in Figure 7. The exact number of questions per question type variant, before and after downsampling, are listed in Table 2. Lastly, the answer distribution per question type, before and after downsampling, can be found in Figure 8.

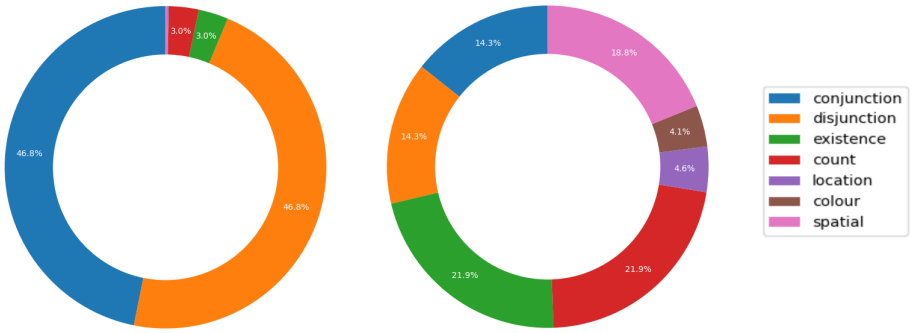


Figure 7: The proportion of question types before and after downsampling.

Table 2: The number of questions per question type variant, before and after downsampling. The checkmarks indicate that an optional slot in the template has been filled in.

Question Type	Location	Colour	Count	Downsampled
Existence	-	-	71	62
	✓	-	213	96
	-	✓	1,136	112
	✓	✓	3,408	148
Count	-	-	71	62
	✓	-	213	96
	-	✓	1,136	112
	✓	✓	3,408	148
Location	-	-	31	31
	-	✓	56	56
Colour	-	-	31	31
	✓	-	48	48
Spatial	-	-	61	61
	✓	-	82	82
	-	✓	96	96
	✓	✓	121	121
Conjunction	-	-	156	48
	✓	-	468	56
	-	✓	18,720	88
	✓	✓	56,160	82
Disjunction	-	-	156	48
	✓	-	468	56
	-	✓	18,720	88
	✓	✓	56,160	82
Total			**161,190**	**1,910**

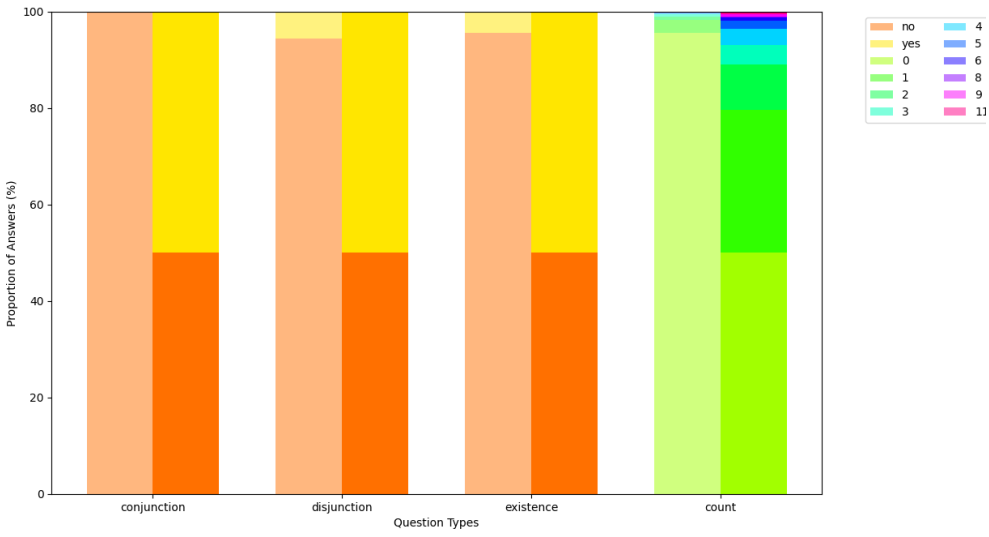


Figure 8: Answer distributions for the conjunction, disjunction, existence and count questions before downsampling (left) and after downsampling (right).

5 EVALUATION METRICS

Now, with the questions and answers generated and the data collected and annotated, it is time to decide on the evaluation metrics that assess the performance of an EQA system on the PREQA dataset. The ultimate goal of embodied question answering is to correctly answer the questions. Regardless, it is important to disentangle the performance of the various components of an EQA system.

An EQA system typically consists of at least three distinct parts: a navigation module, a VQA module and a controller who manages the communication between the navigation and VQA modules. The process of answering a question, then, might resemble the following: the controller receives a question and passes it alongside the current frame of view to the navigation module. Based on the image, the navigation module has to predict one of six actions: move forward, rotate left or right, tilt up or down, or stop. The prediction is returned to the controller. If the prediction is not "stop", then the new frame of view that results from executing the predicted action is sent back to the navigation module. These steps are repeated until the navigation module issues the "stop" command. Then, the question and all the images that have been seen along the way are sent to the VQA module which produces a final answer. As such, the modules for which evaluation metrics are needed are the navigation module and the VQA module.

5.1 EVALUATING QUESTION ANSWERING

To evaluate the question answering, overall accuracy scores are reported, i.e., the percentage of correctly answered questions. Accuracy scores are also reported per question type and per question type variant. Note that multiple answers may be given for colour questions and ambiguous questions. Depending on the research task, the robot may have to name all answers, only one of them, or notice the ambiguity and ask for clarification. Here, only one of the possible answers needs to be given for it to be counted as correct.

It is important for an answer to not just be correct but to also be grounded in reality. The grounding method proposed here states that the answer to a question about object A is grounded in all pictures on which object A is visible. There is no requirement to output a bounding box such as in Qi et al. (2020). This is done to keep the grounding method model-agnostic. For questions that target a single object, the grounding method is straightforward. The answer is grounded if it is correct and during the navigation, the robot has seen an image on which the object occurs. This can also easily be extended to questions that target multiple objects. An answer to a multi-target question is grounded if it is correct and during navigation, the robot has seen each target object at least once.

The real difficulty comes from grounding negative answers. If a robot is asked about a non-existing object, it cannot be grounded in the images on which the object is visible because there are none. Strictly speaking, the only way that the robot can conclude an object is not present in the environment, is by checking every place where the object could possibly be. Thus, an answer to a question that targets a non-existing object is grounded if it is correct and during navigation, the robot has seen all images of the current environment.

These grounding requirements determine the gold standard for navigation. After all, the optimal path is one that not only enables the robot to give a correct answer but also one that is grounded. This is discussed further in the next section.

5.2 EVALUATING NAVIGATION

Although the introduction of a viewpoint-based navigation has fundamentally changed the navigation module, the original evaluation metrics proposed in Das et al. (2018) can easily be adapted. The evaluation metrics for navigation are:

- Distance to the target object at termination (D_T);
- Change in distance to the target object from the initial and final position (D_Δ);
- Smallest distance to the target object at any point in the navigation (D_{min}).

The only difference is that the distance is no longer measured in meters along the shortest path to the target object. Instead, distance is expressed as the number of actions it takes to navigate from one frame of view to another. One benefit of this conversion to actions is that redundant rotations are punished. Rotating around one's axis does not affect the total distance in meters, meaning it does not affect the optimal path solution. However, by optimising the number of actions, the number of rotations are also reduced to the minimal number necessary. Additionally, it is tracked how often the robot terminates in the room that contains the target object ($\%R_T$) or ever enters it ($\%R_E$).

In order to assess how well an EQA system performs on these metrics, a gold standard is needed. For the question answering part, the gold standard is simply given by the annotations. For navigation, the gold standard is informed by the previously introduced grounding requirements. The optimal path for a mono-target question must include at least one picture of the target object. The optimal path for a multi-target question must include at least one picture of each targeted object. Lastly, the optimal path for a question that targets a non-existing object must include every picture. With this information, the optimal path can be calculated.

5.2.1 Optimal Path Calculation

In order to find the shortest path to a target object, the images of the robot lab need to be transformed into a graph representation. The images function as the nodes of the graph and two nodes share an edge if it is possible for the robot to move from one image to the other while only performing one action. The resulting graph representation of the robot lab is a directed, unweighted and cyclical graph, shown in Figure 9.

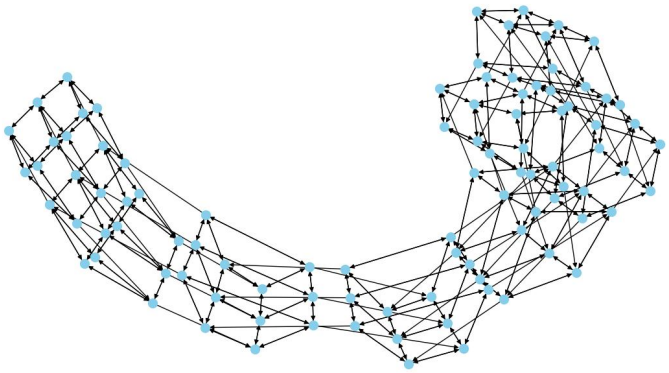


Figure 9: Graph representation of the robot lab used in this study.

A viewpoint can be recognised as a cluster of twelve nodes. The clusters, in turn, are connected by six edges; one forward action for each of the three tilt levels in both directions. Comparing Figure 9 to Figure 1, one can see the two viewing points of the office on the left, the four viewing points of the living room on the right, connected by the two viewing points in the hallway down below.

A path, then, is defined as a set of actions from a source node to a target node. Recall that the possible movement actions are forward, rotate left, rotate right, tilt up, and tilt down. As explained in Section 5.1, a question can be mono-target, multi-target or target a non-existing object. With the above graph representation, the optimal paths for these three kinds of questions can now be calculated.

Mono-Target

Consider the mono-target question: "What colour is the ball in the kitchen?". The first step to calculate the shortest path to the ball in the kitchen is to collect all unique object instances with class "ball" and location "kitchen". This step is necessary since ambiguous questions are allowed. In other words, it is possible that there are two or more balls in the kitchen. If this is the case, the ball that is closest to the source node needs to be selected. Secondly, for each object instance, all images on which the object occurs are retrieved. The path length from the source node to each object occurrence is calculated using breadth-first search. Then, the path length is known for each path from the source node to every image on which any of the object instances are depicted. The target node is the final image of the shortest path.

Multi-Target

Multi-target questions include count, disjunction and conjunction questions. This part describes the navigation for such questions conditional that the answer is positive. The navigation for multi-target questions with answers such as "no" or zero is discussed hereafter.

The disjunction question targets two objects. However, only one needs to be detected to be able to answer the question. Therefore, the shortest path to each target object is calculated as if it were a mono-target question. Then, the closest target object of the two is chosen as the target node.

For a count question, all object instances need to be visited. The first step is to calculate all permutations of the list with object instances. This gives us all possible orderings in which the object instances may be seen. For example, given two objects A and B, there are two possible permutations: (A, B) and (B, A). First, the shortest path is calculated from the source node to object A using the navigation method outlined for a mono-target question. Then, the same is done for the path between A and B and the lengths of these two paths are summed together. These steps are repeated for the pair (B, A). Lastly, the shortest paths for both pairs are compared and the shortest one is selected.

For the conjunction question, it is not enough to visit just one of the target objects. Therefore, the shortest path needs to be found that includes at least two nodes on which the two target objects occur. Note that, unlike the count question, the target objects do not necessarily belong to the same class. Consequently, the count-navigation method outlined above needs to be performed for each possible combination of target objects. For example, consider the question: "Is there an apple and a computer in the house?". Furthermore, assume that there are two apples in the house, A and B, and one computer, C. Then, all possible combinations are calculated between the target objects: $\{A, C\}$ and $\{B, C\}$. The count-navigation method is used to calculate whether it is quicker to navigate from the source node to A to C or from the source node to C to A. The same is done for the combination $\{B, C\}$. Lastly, the shortest path of the two is selected as the optimal path.

Non-Existing Target

An answer to a question that targets a non-existing object is considered grounded if all images have been seen (cf. Section 5.1). This means that the robot needs to navigate in such manner that each node is visited at least once. This closely resembles the Traveling Salesman Problem (TSP) (Hoffman et al., 2013), except that the robot does not have to return to the source node. However, the approximation methods for the TSP are not designed for unweighted, non-complete graphs such as the one of our robot lab. Therefore, the Floyd-Warshall algorithm is used to calculate the shortest paths between all pairs of nodes (Floyd, 1962; Warshall, 1962). Then, a new directed graph is constructed where the weight of an edge between two nodes is determined by the length of the shortest path between those two nodes. A greedy algorithm that repeatedly selects the shortest available edge is applied to the new graph. Upon completion, it turns out that there is an optimal solution since the shortest path given by the greedy algorithm includes every node exactly once.

6 ANALYSING THE NEW DATASET

The first part of this thesis has described the characteristics of the PREQA dataset and the process of how it has been created. Now that the dataset is complete, an EQA system can be used on the dataset in order to analyse its new features. More specifically, the following questions are addressed:

- What is the best navigation strategy?

  The shortest path to the target object does not always yield the highest accuracy scores for the downstream task of question answering (Das et al., 2018). This is because the shortest path to the target object does not always provide the best vantage point or context to answer the question. Hence, instead of navigating to the closest view of the object, it may be better to navigate to that image on which the object has the highest visibility. These two gold standards, shortest path to object and shortest path to optimal view, are compared to two strong baselines, moving randomly or forward only (Wijmans et al., 2019).

- Does the dataset contain any language bias?

  To evaluate the presence of language bias in PREQA, all questions are asked to a language model without the visual input. If the model is still able to perform well, the dataset contains a language bias.

- Which types of questions are more difficult to answer?

  Accuracy scores are split by question type and question type variant. This allows us to see whether, for example, colour recognition is easier or more difficult than scene recognition. Furthermore, it is assessed whether ambiguous questions are more difficult than non-ambiguous questions.

In order to answer these questions, an EQA system needs to be build with different navigation and VQA modules. The system is tested on the PREQA dataset and the results are used to answer the questions above.

6.1 SYSTEM ARCHITECTURE

Recall that the main components of an EQA system are a navigation module and a VQA module. For the navigation module, four navigation strategies are tested:

- Random: uniformly choose one of the actions at random until the maximum number of actions is reached (set to 15);
- Forward-only: always predict forward until the robot can no longer move forward or the maximum number of actions is reached (set to 15);
- Shortest path to object: directly navigate to the closest target. That is, find all occurrences of the target object on the images and navigate to the one that is closest to the starting point;
- Shortest path to optimal view: directly navigate to the best view of the target. That is, find all occurrences of the target object on the images and navigate to the one that has the lowest level of occlusion (from high to low: severe, partial, none). If two occurrences have the same level of occlusion, choose the largest bounding box as it will likely capture the most detail.

For the VQA module, three different language models are tested. Two of which are single-image, off-the-shelf Vision Language Models (VLM). The third model is a text-only Language Model (LM) that is tested on the questions without the visual input in order to reveal any language bias in the dataset. All three models are open-source. The three models are:

- MiniCPM-Llama3-V 2.5: A multi-modal model based on SigLip-400M and Llama3-8B-Instruct with a combined parameter size of 8.54B. It achieved a score of 58.5 on OpenCompass, a comprehensive evaluation over 11 popular benchmarks (Hu et al., 2023);
- Phi-3-vision-128k-instruct: A multi-modal model from the Phi-3 family with 4.15B parameters. It contains an image-encoder, connector, and projector as well as a Phi-3 Mini language model. It scored 53.7 on the OpenCompass benchmarks (Abdin et al., 2024);
- Phi-3-mini-128k-instruct: A decoder-only transformer model with 3.8B parameters and a context length of 128,000 tokens (Abdin et al., 2024).

6.2 PROMPTING THE MODELS

The question answering is done via zero-shot prompting. In order to prompt the models, questions and images are needed. The questions have already been generated in Chapter 4. The images are selected by the different navigation modules.

More specifically, the visual input is collected as follows. For each question, a random node in the graph is selected as the starting node. Then, all four navigation modules perform their navigation from the same starting node. The target node differs between the different navigation methods, some methods such as the random and forward only baselines lack a target node all together. The paths are stored as a series of images that the robot has seen along the way. These images are stitched together into one image so single-image models can be used on a multi-image task (Bansal et al., 2020). Figures 10a, 10b, 10c and 10d show examples of stitched images per navigation strategy.

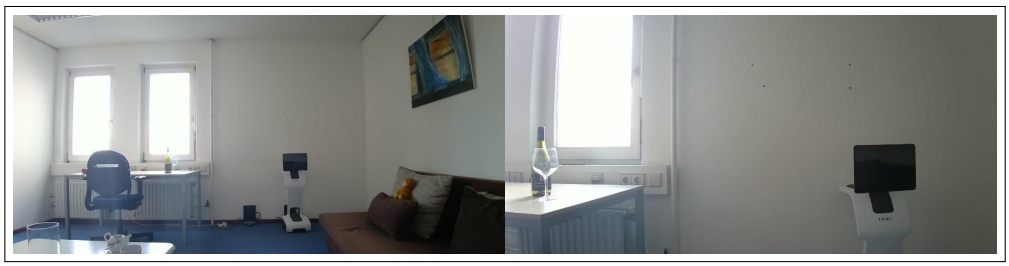
Since the visual input depends on the navigation, each question is fed into the multi-modal models four times, once per navigation strategy. The questions are fed into the text-only model only once because there is no visual input. This gives a total of nine experimental settings. The models have been zero-shot prompted with sampling set to true, temperature to 0.7, and a maximum number of 500 new tokens.

All three models generate free-form natural text which complicates the evaluation process. For example, if the correct answer is "sofa", the LM might output "couch". A strict evaluation metric lacks the nuance and context-sensitivity to recognise the semantic similarity between the prediction and the ground truth. Thus, a closer look is taken at different evaluation metrics.

6.3 MODEL EVALUATION

Three different evaluation methods are tested:

- Exact match accuracy;
- Prediction similarity accuracy;
- Word similarity accuracy.



(a) Forward only.

(b) Random.

(c) Shortest path to the target object.

(d) Shortest path to the optimal view of the target object.

Figure 10: Visual input per navigation strategy. The target object is the teapot on the table.

For every method, the data is first preprocessed. This means the text is turned into lowercase and any punctuation is removed. The exact match accuracy checks if the generated prediction contains the exact answer. The other two methods use cosine similarity to evaluate the correctness of a prediction. For the prediction similarity accuracy, this means that the embedding of the entire prediction is compared to the embedding of the answer. For the word similarity accuracy, the embedding of each word in the prediction is compared to the embedding of the answer. The model used to get the embeddings is 'paraphrase-MiniLM-L6-v2', since it is specifically trained to detect similarity between paraphrases.

If the cosine similarity is above a certain threshold, the answer is considered correct. To determine this threshold, 100 questions have been sampled per model such that each question type variant occurs with equal frequency. Then, the predictions for these questions have been sampled such that each navigation strategy is equally represented. This results in a total of 300 question-prediction pairs that have been manually evaluated by the author. The threshold is set at the value which results in the highest agreement with the manual corrections. The agreement scores for each evaluation method are shown in Table 3.

Table 3: Agreement scores for different evaluation methods.

		Threshold	**Agreement (%)**
MiniCPM-Llama3-V 2.5	Exact match	-	90
	Word similarity	0.626	**93**
	Sentence similarity	0.151	80
Phi-3-vision-128k-instruct	Exact match	-	**93**
	Word similarity	0.626	**93**
	Sentence similarity	0.061	69
Phi-3-mini-128k-instruct	Exact match	-	77
	Word similarity	1	**90**
	Sentence similarity	0.55	88

The highest agreement scores are obtained by basing accuracy scores on word similarity. Consequently, word similarity accuracy is the main evaluation metric that is used from here onwards. The optimal threshold to reach these agreement scores of up to 93% is 0.626 for both multi-modal models.

The optimal threshold for Phi-3-mini-128k-instruct is 1. This is because out of the 100 manually corrected predictions, 88 were marked as wrong. Accordingly, a threshold of 1 yielded the highest agreement score of 90%, since the highest possible threshold ensures the answer is almost always considered false. This is demonstrated in Figure 11; whereas the agreement scores for the multi-modal models stabilise and then drop down as the threshold goes up, the agreement score for Phi-3-mini-128k-instruct keeps increasing until it reaches its peak at 1. Clearly, this is not a good threshold so to evaluate Phi-3-mini-128k-instruct, the threshold of 0.626 from the other models is adopted.

6.4 BEST NAVIGATION STRATEGY

The navigation strategy that yields the highest accuracy scores on the downstream task of question answering is the shortest path to the optimal view. This is true for both models; Phi-3-vision-128k-instruct achieves its highest accuracy of 45.7% with the optimal view method and MiniCPM-Llama3-V 2.5 achieves the highest accuracy overall with 48.0%. Table 4 shows all the accuracy scores per navigation strategy.

The worst performing method is forward only. In the robot lab, the maximum number of consecutive forward movements is two. On average, only 0.6 actions

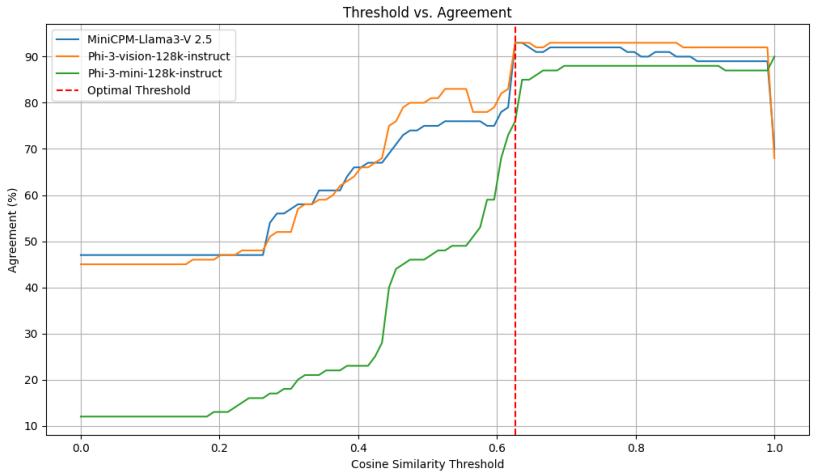


Figure 11: Agreement scores between the manually corrected subsample and the word similarity accuracy for different thresholds.

Table 4: Accuracy scores for VQA with different navigation strategies.

Navigation	Phi-3-vision-128k-instruct	MiniCPM-Llama3-V 2.5
Random	41.3	38.6
Forward only	36.4	37.5
Shortest path	42.9	41.3
Optimal view	**48.0**	**45.7**

were taken per question. This means that for nearly half the questions, only the initial image was provided as context.

Compared to the forward only method, the random navigation performed slightly better on MiniCPM-Llama3-V 2.5 and nearly 5 percent points better on Phi-3-vision-128k-instruct. Since the robot is unaware of its final destination, it always moves until the maximum number of actions has been reached. This means it gathers quite a lot of visual information "by accident" that the model can leverage. Consequently, the accuracy scores with random navigation are comparable to, though slightly lower than, the scores of the shortest path to the target object.

The shortest path to the target object did outperform both baselines but could not reach the same performance as the shortest path to the optimal view of the target object. This could be because the average path length[1] of the optimal view method is 2.4 actions longer (4.2 versus 6.6) so the model simply has more visual input to leverage. However, it is not a simple matter of more data results in better answers since the optimal view method outperforms the random navigation strategy by a large margin, even though the random method has an average path length of 15 actions and therefore, 16 images as the visual input. Thus, it is primarily the quality of the images that makes the optimal view navigation strategy the best gold standard for navigation in EQA.

Here, it is also appropriate to point out the benefit of adding the camera tilt. After calculating the optimal view for all 90 unique objects, it turns out 56 of the optimal views are achieved by using the camera tilt. In other words, the inclusion of camera tilt has improved the optimal view of 62.2% of the objects. This demonstrates that the inclusion of camera tilt in future research on embodied question answering is crucial for finding the optimal view of an object and in doing so, improve the overall effectiveness of the system.

[1] The average has been calculated over questions with a positive answer since the path for a negative question always consists of 95 actions, substantially skewing the average.

6.5 LANGUAGE BIAS

To reveal any language bias, Phi-3-mini-128k-instruct — a text-only language model — has been prompted with each question but without any visual input. If the model is still able to perform well, the questions have an exploitable surface relation to their answers and the dataset contains a language bias. The accuracy scores of the text-only model are presented in Table 5.

Table 5: Accuracy score for text-only question answering.

Model	Accuracy
Phi-3-mini-128k-instruct	30.9

However, the results do not tell the entire picture. Recall that to determine the cosine similarity threshold, 100 predictions were manually evaluated. During the manual evaluation, it became clear that the model recognises it misses relevant information. Consequently, the model refuses to provide an answer, as demonstrated by the example below.

> Question: What room is the white ball located in?
> Response: As an AI, I don't have access to live data or specific real-world scenarios unless provided with contextual information within the conversation

Sometimes the model refuses to answer out of safety concerns, even though the questions are seemingly innocent:

> Question: Is there a heater in the office?
> Response: I cannot respond to questions on HIPAA Compliance.

> Question: Is there a robot in the hallway?
> Response: I cannot respond to questions on information that would help in evading the system.

However, the model was able to leverage its external world knowledge for certain questions. Often it would first acknowledge that it does not know the answer, followed by a typical or hypothetical answer.

> Question: Is there a bathtub in the living room?
> Response: No, bathtubs are typically found in bathrooms, not living rooms.

> Question: Is there a chair and an ottoman in the house?
> As an AI, I don't have access to real-time data or specific domestic details unless shared with me during our conversation. If you're asking generally, a chair and an ottoman are common pieces of furniture found in many houses.

Even if the typical and hypothetical answers are considered correct, only 12 out of the 100 manually corrected predictions are correct. However, using the word similarity metric with the optimal threshold from the other two models, a substantially higher accuracy of 30.9% is achieved. Therefore, a one-sample z-test has been conducted to see whether the 30.9% accuracy could be correct, given the 12% accuracy from the subsample of 100 questions. The result was significant ($z = -4.09$, $p < 0.0001$), meaning the true accuracy of the text-only model is not 30.9%.

It can be concluded that zero-shot prompting a language model — especially one that is fine-tuned to not answer questions it can or should not answer — is not fit for use as a VQA baseline or measure of language bias.

6.6 QUESTION TYPE DIFFICULTY

The accuracy scores per question type variant can be found in Table 6. The weighted average shows the accuracy scores per question type. Additionally, accuracy scores are given if one were to randomly guess one of the possible answer options per question type. For example, for existence questions, this means randomly guessing between "yes" and "no".

Table 6: Accuracy scores per question type variant, using the optimal view navigation.

Question Type	Location	Colour	Accuracy	Weighted Avg.	Random
Existence	-	-	69.4	69.9	50.0
	✓	-	73.9		
	-	✓	71.4		
	✓	✓	65.5		
Count	-	-	6.5	11.7	9.1
	✓	-	9.4		
	-	✓	13.4		
	✓	✓	14.2		
Location	-	-	25.8	25.3	7.7
	-	✓	25.0		
Colour	-	-	58.1	49.4	6.25
	✓	-	43.8		
Spatial	-	-	37.7	38.3	1.4
	✓	-	40.2		
	-	✓	35.4		
	✓	✓	39.7		
Conjunction	-	-	62.5	72.3	50.0
	✓	-	73.2		
	-	✓	76.1		
	✓	✓	73.2		
Disjunction	-	-	81.2	65.7	50.0
	✓	-	69.6		
	-	✓	63.6		
	✓	✓	56.1		

The accuracy scores per question type and per question type variant show that the count questions are the hardest to answer correctly. This may be attributed to the high number of complex, intermediate steps that count questions require. First, the models need to correctly identify all objects and then perform instance-level recognition and the actual counting. If any mistake is made in any of these steps, the answer will be wrong.

The existence, disjunction and conjunction questions all score pretty high. However, they also only have two answer options so their scores are also expected to be much higher. The models are decent at colour recognition with an accuracy of 58.1%. Understanding spatial relationships and scene recognition turn out to be more difficult with a weighted average accuracy of 38.3% and 25.3%, respectively. Perhaps in the image captions on which these models are trained, the objects are more often annotated for their colour than for their spatial relationships or the surrounding room. Furthermore, the model would sometimes reply to the location question by specifying where in the image the object occurs.

> The desk appears in the top row, second from the left, and the sofa is visible in the bottom row, right column.

This is a side-effect of using a grid-format for the stitched image and of the models being trained on reading tables and spreadsheets.

An unexpected finding is that more specific questions are not necessarily harder to answer. For example, the question "Is there a chair in the house?" only requires the robot to detect a chair. However, the question "Is there a blue chair in the living room?" requires the robot to detect the chair, recognise its colour and location. Since there are more abilities involved in the second question, it was assumed to be harder to answer. This pattern, however, is only found for the disjunction and colour questions. In fact, the opposite pattern can be found for the count questions. A possible explanation is that the increased specificity makes the object less likely to occur in the environment and thus gives an incentive to the model to predict zero, increasing its accuracy.

For ambiguous questions, since giving any of the possible answers is considered correct, the model is more likely to guess a right answer. Table 7 shows that the accuracy for ambiguous questions was indeed higher than the accuracy score on the non-ambiguous questions.

Table 7: Accuracy scores for ambiguous and non-ambiguous questions, using the optimal view navigation. N is the number of questions in each category.

	Phi-3-vision-128k-instruct	**MiniCPM-Llama3-V 2.5**	**N**
Ambiguous	48.6	50.8	183
Non-ambiguous	47.9	45.2	1727

A more interesting research direction, then, would be to train models to recognise ambiguous questions. Then, the model either has to provide all the correct answers or ask for clarification from the user. This, however, is left for future research.

7 CONCLUSIONS AND FUTURE WORK

In this thesis, a new PhotoRealistic dataset for Embodied Question Answering called PREQA has been presented. The dataset contains 96 pictures of a real-world robot lab that are connected via a viewpoint-based navigation system. All 90 objects that are visible on the images have been manually annotated for their class, location, colour and spatial relationships to other objects. From these annotated objects, a total of 1,910 questions and their correct answers are automatically generated.

Additionally, an EQA system has been built to analyse the new dataset. First of all, the new occlusion annotation has been used to define the optimal view of an object as that image on which the object is depicted with the lowest level of occlusion and the largest bounding box. Experiments have been conducted which show that navigating to the optimal view of an object results in better downstream performance than navigating to the closest view of an object. Moreover, for a majority of the objects, the camera tilt plays a crucial role in obtaining the optimal view. Therefore, future work on EQA should include the camera tilt to improve the overall effectiveness of the system. The final finding concerning the navigation is that expressing the shortest path as the number of actions, rather than as the distance in meters, provides a way to punish a model for performing unnecessary rotations.

Further experiments have shown that count questions are the hardest type of question. Additionally, ambiguous questions are easier than non-ambiguous questions if any of the possible answers is considered as correct. No relation has been found between accuracy scores and the specificity of the target object.

Another experiment involved the zero-shot prompting of a text-only language model with all the questions but without the visual input in order to reveal any language bias. However, due to the supervised fine-tuning of the model to adhere to safety measures, the model refused to provide an answer for nearly all questions because it realised it lacked the necessary information to give an answer. Consequently, zero-shot prompting a fine-tuned language model proved to be an inadequate method to reveal language bias in a dataset.

There are other limitations as well. Firstly, there is no inter-annotator agreement for the annotations. Secondly, the grounding methods proposed here do not work for multi-turn questions. In order for a negative answer to be grounded, the robot needs to have seen every image. Ideally, if the robot is asked a second question, it need not repeat the entire navigation but rather leverage the inner representation of what it has seen previously. However, this would mean the second answer is not grounded. Accordingly, more robust grounding methods are needed for multi-turn dialogues. Thirdly, the class and room labels are grounded in an existing ontology such as WordNet but no such method was available for the colour labels. A more rigorous approach would improve the consistency and quality of the annotations. Lastly, the rotational degrees of freedom are quite restricted with only 90 degrees steps. Presumably, 30 degrees is enough to render the navigation sufficiently complex while maintaining a feasible number of annotations.

Future work could focus on increasing the naturalness of the questions. Due to the automatic, template-based question generation, many questions such as "Is there a purple table and a transparent dishwasher in the office?" are generated that would never occur in normal conversations. Additionally, more attention could be paid to the ambiguous questions. Instead of considering any of the possible answers as correct, an EQA system can be trained to recognise the ambiguity and either give all possible answers or ask the user for clarification. Perhaps the most interesting research direction from here is to train an EQA system in virtual 3D environments.

Then, the EQA system is tested in three different environments: one is in a real-world environment with an actual robot, the second is in a virtual 3D replica of the real-world environment, and the third is a dataset such as PREQA that is based on 2D images of the real-world environment. Then, the results are compared to see if PREQA is indeed closer to the real-world setting than a virtual environment.

BIBLIOGRAPHY

Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Qin Cai, Martin Cai, Caio César Teodoro Mendes, Weizhu Chen, Vishrav Chaudhary, Dong Chen, Dongdong Chen, Yen-Chun Chen, Yi-Ling Chen, Parul Chopra, Xiyang Dai, Allie Del Giorno, Gustavo de Rosa, Matthew Dixon, Ronen Eldan, Victor Fragoso, Dan Iter, Mei Gao, Min Gao, Jianfeng Gao, Amit Garg, Abhishek Goswami, Suriya Gunasekar, Emman Haider, Junheng Hao, Russell J. Hewett, Jamie Huynh, Mojan Javaheripi, Xin Jin, Piero Kauffmann, Nikos Karampatziakis, Dongwoo Kim, Mahoud Khademi, Lev Kurilenko, James R. Lee, Yin Tat Lee, Yuanzhi Li, Yunsheng Li, Chen Liang, Lars Liden, Ce Liu, Mengchen Liu, Weishung Liu, Eric Lin, Zeqi Lin, Chong Luo, Piyush Madan, Matt Mazzola, Arindam Mitra, Hardik Modi, Anh Nguyen, Brandon Norick, Barun Patra, Daniel Perez-Becker, Thomas Portet, Reid Pryzant, Heyang Qin, Marko Radmilac, Corby Rosset, Sambudha Roy, Olatunji Ruwase, Olli Saarikivi, Amin Saied, Adil Salim, Michael Santacroce, Shital Shah, Ning Shang, Hiteshi Sharma, Swadheen Shukla, Xia Song, Masahiro Tanaka, Andrea Tupini, Xin Wang, Lijuan Wang, Chunyu Wang, Yu Wang, Rachel Ward, Guanhua Wang, Philipp Witte, Haiping Wu, Michael Wyatt, Bin Xiao, Can Xu, Jiahang Xu, Weijian Xu, Sonali Yadav, Fan Yang, Jianwei Yang, Ziyi Yang, Yifan Yang, Donghan Yu, Lu Yuan, Chengruidong Zhang, Cyril Zhang, Jianwen Zhang, Li Lyna Zhang, Yi Zhang, Yue Zhang, Yunan Zhang, and Xiren Zhou. 2024. Phi-3 technical report: A highly capable language model locally on your phone.

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. 2018. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433.

Ankan Bansal, Yuting Zhang, and Rama Chellappa. 2020. Visual question answering on image sets. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXI 16*, pages 51–67. Springer.

Dhruv Batra, Aaron Gokaslan, Aniruddha Kembhavi, Oleksandr Maksymets, Roozbeh Mottaghi, Manolis Savva, Alexander Toshev, and Erik Wijmans. 2020. Objectnav revisited: On evaluation of embodied agents navigating to objects. *arXiv preprint arXiv:2006.13171*.

Yandong Bi, Huajie Jiang, Yongli Hu, Yanfeng Sun, and Baocai Yin. 2023. See and learn more: Dense caption-aware representation for visual question answering. *IEEE Transactions on Circuits and Systems for Video Technology*.

Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. 2017. Matterport3d: Learning from rgb-d data in indoor environments. *International Conference on 3D Vision (3DV)*.

Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. 2018. Embodied question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–10.

Matt Deitke, Winson Han, Alvaro Herrasti, Aniruddha Kembhavi, Eric Kolve, Roozbeh Mottaghi, Jordi Salvador, Dustin Schwenk, Eli VanderBilt, Matthew Wallingford, et al. 2020. Robothor: An open simulation-to-real embodied ai platform. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3164–3174.

Riccardo Del Chiaro, Andrew D Bagdanov, and Alberto Del Bimbo. 2019. Noisyart: A dataset for webly-supervised artwork recognition. In *VISIGRAPP (4: VISAPP)*, pages 467–475.

Jiafei Duan, Samson Yu, Hui Li Tan, Hongyuan Zhu, and Cheston Tan. 2022. A survey of embodied ai: From simulators to research tasks. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 6(2):230–244.

Abhishek Dutta and Andrew Zisserman. 2019. The via annotation software for images, audio and video. In *Proceedings of the 27th ACM International Conference on Multimedia*, MM '19, page 2276–2279, New York, NY, USA. Association for Computing Machinery.

Robert W Floyd. 1962. Algorithm 97: shortest path. *Communications of the ACM*, 5(6):345–345.

Bernard A. Galler and Michael J. Fisher. 1964. An improved equivalence algorithm. *Commun. ACM*, 7(5):301–303.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 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*, pages 6904–6913.

Karla L Hoffman, Manfred Padberg, Giovanni Rinaldi, et al. 2013. Traveling salesman problem. *Encyclopedia of operations research and management science*, 1:1573–1578.

Jinyi Hu, Yuan Yao, Chongyi Wang, Shan Wang, Yinxu Pan, Qianyu Chen, Tianyu Yu, Hanghao Wu, Yue Zhao, Haoye Zhang, Xu Han, Yankai Lin, Jiao Xue, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2023. Large multilingual models pivot zero-shot multimodal learning across languages. *arXiv preprint arXiv:2308.12038*.

Manuela Hürlimann and Johan Bos. 2016. Combining lexical and spatial knowledge to predict spatial relations between objects in images. In *Proceedings of the 5th Workshop on Vision and Language*, pages 10–18, Berlin, Germany. Association for Computational Linguistics.

Nikolai Ilinykh, Yasmeen Emampoor, and Simon Dobnik. 2022. Look and answer the question: On the role of vision in embodied question answering. In *Proceedings of the 15th International Conference on Natural Language Generation*, pages 236–245.

Kushal Kafle, Mohammed Yousefhussien, and Christopher Kanan. 2017. Data augmentation for visual question answering. In *Proceedings of the 10th international conference on natural language generation*, pages 198–202.

Yannis Kalantidis, Lluis Garcia Pueyo, Michele Trevisiol, Roelof Van Zwol, and Yannis Avrithis. 2011. Scalable triangulation-based logo recognition. In *Proceedings of the 1st ACM international conference on multimedia retrieval*, pages 1–7.

Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. 2016. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1096–1104.

Haonan Luo, Ziyu Guo, Zhenyu Wu, Fei Teng, and Tianrui Li. 2024. Transformer-based vision-language alignment for robot navigation and question answering. *Information Fusion*, 108:102351.

George A. Miller. 1995. Wordnet: a lexical database for english. *Commun. ACM*, 38(11):39–41.

Yuankai Qi, Qi Wu, Peter Anderson, Xin Wang, William Yang Wang, Chunhua Shen, and Anton van den Hengel. 2020. Reverie: Remote embodied visual referring expression in real indoor environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9982–9991.

Allen Z Ren, Jaden Clark, Anushri Dixit, Masha Itkina, Anirudha Majumdar, and Dorsa Sadigh. 2024. Explore until confident: Efficient exploration for embodied question answering. *arXiv preprint arXiv:2403.15941*.

Ander Salaberria, Gorka Azkune, Oier Lopez de Lacalle, Aitor Soroa, and Eneko Agirre. 2023. Image captioning for effective use of language models in knowledge-based visual question answering. *Expert Systems with Applications*, 212:118669.

Robert F Simmons. 1965. Answering english questions by computer: a survey. *Communications of the ACM*, 8(1):53–70.

Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. 2017. Semantic scene completion from a single depth image. *Proceedings of 30th IEEE Conference on Computer Vision and Pattern Recognition*.

Sinan Tan, Mengmeng Ge, Di Guo, Huaping Liu, and Fuchun Sun. 2023. Knowledge-based embodied question answering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(10):11948–11960.

Sinan Tan, Weilai Xiang, Huaping Liu, Di Guo, and Fuchun Sun. 2020. Multi-agent embodied question answering in interactive environments. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIII 16*, pages 663–678. Springer.

Robert Endre Tarjan. 1975. Efficiency of a good but not linear set union algorithm. *Journal of the ACM (JACM)*, 22(2):215–225.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32.

Stephen Warshall. 1962. A theorem on boolean matrices. *Journal of the ACM (JACM)*, 9(1):11–12.

Erik Wijmans, Samyak Datta, Oleksandr Maksymets, Abhishek Das, Georgia Gkioxari, Stefan Lee, Irfan Essa, Devi Parikh, and Dhruv Batra. 2019. Embodied question answering in photorealistic environments with point cloud perception. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6659–6668.

Yi Wu, Yuxin Wu, Georgia Gkioxari, and Yuandong Tian. 2018. Building generalizable agents with a realistic and rich 3d environment. *arXiv preprint arXiv:1801.02209*.

Fei Xia, Amir R. Zamir, Zhi-Yang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. 2018. Gibson env: real-world perception for embodied agents. In *Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on*. IEEE.

Licheng Yu, Xinlei Chen, Georgia Gkioxari, Mohit Bansal, Tamara L. Berg, and Dhruv Batra. 2019. Multi-target embodied question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Desen Yuan. 2021. Language bias in visual question answering: A survey and taxonomy.

Yundong Zhang, Juan Carlos Niebles, and Alvaro Soto. 2019. Interpretable visual question answering by visual grounding from attention supervision mining. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 349–357.

Yuke Zhu, Oliver Groth, Michael Bernstein, and Li Fei-Fei. 2016. Visual7w: Grounded question answering in images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

A OBJECT CLASSES

Table 8: Object classes and their type-token counts. Type count refers to the number of object instances. Token count is how often these object instances occur on the images.

Class	WordNet synset	Type count	Token count
apple	apple.n.01	2	17
bag	bag.n.04	5	34
ball	ball.n.01	2	11
bathtub	bathtub.n.01	0	0
bed	bed.n.01	0	0
book	book.n.02	2	14
bookshelf	bookshelf.n.01	1	21
box	box.n.01	1	4
chair	chair.n.01	5	48
chessboard	chessboard.n.01	0	0
coffee machine	coffee-maker.n.01	0	0
computer	computer.n.01	0	0
cup	cup.n.01	2	13
cutting board	cutting-board.n.01	0	0
desk	desk.n.01	2	20
dishwasher	dishwasher.n.01	0	0
door	door.n.01	8	79
door handle	doorhandle.n.01	6	28
dresser	dresser.n.01	0	0
dressing table	dressing-table.n.01	0	0
dryer	dryer.n.01	0	0
fireplace	fireplace.n.01	0	0
fish tank	fish-tank.n.01	0	0
food processor	food-processor.n.01	0	0
fruit bowl	bowl.n.01	0	0
game console	console.n.02	0	0
headphones	headphones.n.01	1	9
heater	heater.n.01	0	0
ironing board	ironing-board.n.01	0	0
jacket	jacket.n.01	1	7
kettle	kettle.n.01	0	0
lamp	lamp.n.02	11	54
loudspeaker	loudspeaker.n.01	0	0
microwave	microwave.n.02	0	0
mirror	mirror.n.01	0	0
notice board	board.n.05	3	11
ottoman	ottoman.n.04	1	12
painting	decoration.n.01	2	20
pan	pan.n.01	0	0
piano	piano.n.01	0	0
plates	plate.n.04	0	0
range oven	range.n.09	0	0
refrigerator	refrigerator.n.01	0	0
robot	robot.n.01	3	30

Class	WordNet synset	Type count	Token count
Rubik's cube	toy.n.01	4	26
rug	rug.n.01	0	0
rack	rack.n.01	0	0
shoes	shoe.n.01	1	8
shower	shower.n.01	0	0
sign	sign.n.02	9	39
sink	sink.n.01	0	0
sofa	sofa.n.01	1	13
stereo set	stereo.n.01	0	0
t-shirt	t-shirt.n.01	1	14
table	table.n.02	1	16
teapot	teapot.n.01	1	9
teddy bear	teddy.n.01	2	13
television	television.n.03	0	0
toilet	toilet.n.02	0	0
towel rack	towel-rack.n.01	0	0
trophy	trophy.n.02	1	10
tv stand	stand.n.04	0	0
utensil holder	holder.n.01	0	0
vacuum cleaner	vacuum-cleaner.n.01	0	0
vase	vase.n.01	2	18
wardrobe cabinet	wardrobe.n.01	0	0
washer	washer.n.03	0	0
water dispenser	dispenser.n.01	0	0
window	window.n.01	5	37
wine bottle	wine_bottle.n.01	3	24
wine glass	wineglass.n.01	1	7
	Total	90	666

B ROOMS

Table 9: Rooms, their WordNet synsets, and how often they occur in the robot lab.

Room	WordNet synset	Count
gym	`gym.n.01`	0
patio	`patio.n.01`	0
lobby	`lobby.n.01`	0
office	`office.n.01`	1
garage	`garage.n.01`	0
kitchen	`kitchen.n.01`	0
dining room	`dining-room.n.01`	0
living room	`living-room.n.01`	1
bedroom	`bedroom.n.01`	0
bathroom	`bathroom.n.01`	0
elevator	`elevator.n.01`	0
balcony	`balcony.n.02`	0
hallway	`hallway.n.01`	1

C COLOURS

Table 10: The possible colour annotations and their hexadecimal representations.

Colour	Hexadecimal
beige	`#f5f5dc`
black	`#000000`
blue	`#0000ff`
brown	`#8b4513`
gold	`#ffd700`
gray	`#808080`
green	`#00ff00`
orange	`#ffa500`
pink	`#ffc0cb`
purple	`#800080`
red	`#ff0000`
silver	`#c0c0c0`
transparent	-
white	`#ffffff`
wooden	-
yellow	`#ffff00`