

---

# EMMA: A Foundation Model for Embodied, Interactive, Multimodal Task Completion in 3D Environments

---

Amit Parekh<sup>§\*</sup> Malvina Nikandrou\* Georgios Pantazopoulos\* Bhatiya Hemanthage\*

Arash Eshghi Ioannis Konstas Oliver Lemon Alessandro Suglia<sup>†</sup>

{amit.parekh, mn2002, gmp2000, hsb2000,  
a.eshghi, i.konstas, o.lemon, a.suglia}@hw.ac.uk

## Abstract

In this technical report, we present EMMA, a foundation model for embodied, interactive, and multimodal task completion in 3D environments. Different from previous Vision+Language (V+L) models, EMMA is an encoder-decoder architecture that encodes both images and videos (i.e., sequences of frames), and it is able to generate natural language tokens conditioned on specific *task prompts*. By treating every task as a natural language generation task, EMMA learns a *language of actions* that can be used for different tasks in the pipeline of an embodied AI system. We perform an extensive experimental evaluation to demonstrate the performance of our foundation model. First, despite being substantially smaller than other V+L models, EMMA is competitive (or superior) in terms of performance on several V+L state-of-the-art benchmarks demonstrating the value of our model design and multitask pretraining regime. Additionally, we showcase that a model trained on Alexa Arena data can perform zero-shot cross-domain transfer when asked to perform the same tasks in the real world. Moreover, EMMA shows strong generalization performance in novel missions with real users, achieving an average score of 4.06 (out of 5) over the generalization phase that lasted between the 16<sup>th</sup> and 22<sup>nd</sup> of March 2023.

## 1 Introduction

Language is a fundamental aspect of human communication and a critical component of human-robot collaboration. Humans use language to convey complex information and instructions or pose informative questions and clarifications that minimize task failure. As robots become more advanced, it is essential to develop effective ways for humans to interact and communicate with them. By incorporating language capabilities into robots, they can become more useful and adaptable to different contexts, making them valuable assets in various collaborative tasks. Any advances in improving language understanding for robots have the potential to revolutionize the way we interact with machines and can open up new possibilities for human-robot collaboration.

---

<sup>§</sup>Team Lead

\*Equal Contribution

<sup>†</sup>Lead Faculty Advisor

However, language-driven behavior requires systematic generalization — the ability to combine known representations to address novel scenes, user queries, or tasks [6, 28]. To address this challenge, we propose EMMA, a foundation model for embodied, interactive, and multi-modal task completion that encodes scenes using fine-grained object-centric representations [44]. These representations are learned through a multi-phase pretraining scheme, enabling our agent to capture relevant entities in the environment and learn to reason about them. The multi-phase pretraining aims at developing language grounding capabilities in visual environments suitable for task execution and interaction. Differently from previous V+L models, EMMA is an encoder-decoder architecture that encodes both images and videos (i.e., sequences of frames), and it is able to generate natural language tokens conditioned on specific task prompts.

In the context of the Alexa Prize SimBot Challenge, we have developed an embodied AI agent with EMMA as the core building block. We have fine-tuned EMMA for downstream tasks within the Alexa Arena [12], while also implementing supporting features that facilitate the user experience. Our system interacts with users to solve missions in a scalable, robust, and engaging manner.

## 2 Related Work

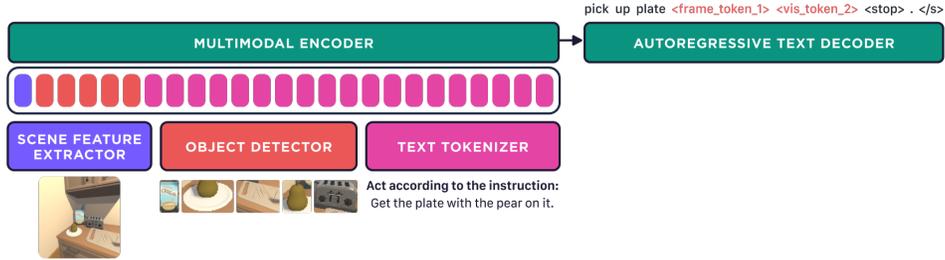
**Vision-and-Language Pretraining** Representation learning with transformer-based models has been increasingly successful in many V+L tasks. Following Bugliarello et al. [9], we can define two types of multimodal architectures: single-stream and dual-stream transformers. Single-stream transformers (e.g., [10, 42, 43]) assume that there is a single transformer stack receiving both visual and textual inputs. Dual-stream transformers (e.g., [2, 21, 24]) instead assume that each modality is encoded by a modality-specific encoder whose outputs are fused by dedicated cross-modal layers. Bugliarello et al. [9] demonstrate that dual-stream attention functions act as restricted versions of the attention function in any single-stream transformer architecture and under controlled pretrained settings models of both architecture types perform comparably.

Recently, there has been increasing interest in models that cast multiple tasks in unified frameworks. Unified V+L frameworks are motivated by advances in Natural Language Processing (NLP), which have demonstrated the possibility of treating a wide range of text-processing problems as a text-to-text problem [31]. Based on these works, unified V+L models typically focus on text generation conditioned on multimodal representations obtained from the visual and textual inputs. To generate outputs that can reference objects within the image, Cho et al. [10] augment the model vocabulary with special visual sentinel tokens to refer to specific objects. Models that encode image patches [42, 43], however, include in the vocabulary quantized location tokens, which can be used to reference bounding boxes in the image. In EMMA, we follow the former approach by equipping our model to generate sentinel tokens which refer to objects in the scene. In this way, we can complete important tasks such as object manipulation.

**Embodied AI Approaches** A variety of benchmarks have been proposed to tackle complex tasks for embodied AI, including visual navigation [4], and language-guided task execution [36]. Previous approaches tackling these benchmarks can be divided into end-to-end and modular architectures.

By end-to-end architectures, we refer to models trained on demonstrations to generate a sequence of actions. For example, Anderson et al. [4] present an attention-based RNN to generate actions for addressing vision-language navigation tasks. Similarly, Shridhar et al. [36] present a model trained in a multitask fashion to predict both actions and the object masks used for object manipulation. Recently, Episodic Transformer [30] proposes an autoregressive transformer model that conditions its action generation on language instructions, past actions, and scene features that encode the environment state. It is important to stress that these models are trained from scratch on a specific environment and therefore struggle to generalize to novel environments and tasks. To mitigate this, recent attempts have adapted pretrained V+L models to embodied AI tasks. Majumdar et al. [25] fine-tune ViLBERT [24] to complete vision-language navigation tasks. Suglia et al. [38] advances this idea by combining the pretrained V+L model OSCAR [22] with a Transformer-XL decoder [11].

Another approach to embodied AI is to compose systems from several standalone modules; a strategy that has performed well on the popular ALFRED challenge [36]. For example, MOCA [37] combines



**Figure 1.** Overview of the architecture for the EMMA Foundation Model.

a policy module trained to predict actions with a vision module trained to predict target objects. Blukis et al. [7] further augment MOCA with a search policy module to help guide the model towards relevant receptacles. The FILM model proposed by Min et al. [27] continues this line of work by introducing a hand-crafted high-level planner to generate a sequence of sub-goals. These solutions achieve high success rates as they rely on assumptions that are potentially only valid in certain environments; for example, the MOCA instance association trick, or the object-slicing rule in FILM.

Compared to more hand-crafted approaches that are benchmark-specific, we consider EMMA as a foundation model for embodied AI. EMMA has the ability to solve multiple tasks from natural language prompts, allowing it to easily adapt to the types of tasks required within an embodied AI solution. Additionally, as it has only 113M parameters, it is easy to run at scale and still provides competitive performance with state-of-the-art models.

### 3 The EMMA Foundation Model

*What are the benefits of a multitask over a conventional model for human-robot collaboration?* By incorporating multiple tasks into one architecture, the agent learns to match textual descriptions of objects to visual cues, answer questions about the scene, and describe it using language. With complementary tasks, this approach can improve overall performance and reduce the need for specialized models for each task, resulting in a more effective system. Additionally, the agent can learn to perform tasks in a more human-like way, leading to better user engagement and interaction.

*What tasks can benefit an embodied agent?* On top of performing both navigation and manipulation actions, a competent embodied AI agent must also simultaneously complete additional tasks, such as visual grounding (i.e., understanding the referent from a given referential expression), visual question answering (i.e., answering questions about a scene), and image captioning.

For these reasons, we designed EMMA: a model that uses multimodal representations from visual data in the form of images, videos, or action trajectories paired with textual data such as question-answer pairs and captions. EMMA is a transformer encoder-decoder trained with natural language prompts [10, 31] to cast every task objective as a form of text generation. We designed a pretraining phase to expose the model to various useful concepts for completing tasks in an embodied environment, such as visual grounding, object manipulation, and scene description. After pretraining, our model can be fine-tuned on downstream tasks, including visual ambiguity detection and action prediction.

#### 3.1 Model Architecture

We use a transformer-based encoder-decoder architecture following BART [20]. As shown in Figure 1, both vision and language inputs are embedded through modality-specific layers, concatenated into a sequence of embeddings, and fed into a single-stream encoder. In contrast to typical encoder-only models that use task-specific classification heads [22, 24], EMMA uses a shared decoder across all pretraining tasks by formulating all tasks as language generation and incorporating sentinel tokens in the vocabulary to allow referencing specific image frames and regions.

**Table 1.** Example input and output formats used for the pretrained tasks.

Task	Example input	Target output
MLM	Denoise: Fridge <MASK> is open	Fridge door is open
ITM	Assess the statement: Fridge door is open	True
Captioning	Describe the image	Food inside a refrigerator with its door open
Dense Captioning	Describe object <visual_token_5>	silver fridge
Visual Grounding	Locate the milk carton	<visual_token_3>
VQA	What color are the cabinets?	White
Relationship Detection	Explain how <visual_token_3> relates to <visual_token_5>	Milk inside of fridge

**Text Embedding** For the language input, we apply sub-word byte-pair encoding [34] with a vocabulary of 10K tokens extracted from our pretraining data. Language tokens  $L = \{w_1, \dots, w_L\}$  are represented by the sum of their word and absolute positional embedding following the approach used by Lewis et al. [20]. We use task prefixes to prompt the model for the various tasks. We experimented with two types of prefixes: task-specific special tokens and natural language prompts. Task-specific special tokens are single-word descriptors (originally proposed in Cho et al. [10]), while natural language prompts are longer, varied descriptions similar to the approach proposed by Sanh et al. [33]. For example, the image captioning task is denoted either by the single token [Cap] or by prompts such as “Describe this” or “Caption the image”. From early experiments, we found that both strategies performed comparably. However, we chose to use natural language prompts because they allowed for a more flexible interface with the model.

**Scene and Object Embeddings** EMMA is capable of encoding sequences of images. For each frame  $I_v$  of the visual input,  $V = \{I_1, \dots, I_V\}$ , we extract a maximum of  $n = 36$  region features  $R_v$  and bounding boxes  $B_v$  using an object detection model. Additionally, we extract global scene features representing the entire frame. The image embedding layers project the visual features to the encoder’s dimensionality and add spatiotemporal positional embeddings. In particular, the positional embeddings encode the 2D spatial position of the region within the frame by its normalized bounding box coordinates and the temporal position within the sequence using a frame sentinel token. Finally, to reference each object in the image, the language vocabulary is extended with visual sentinel tokens in the form of <visual\_token\_ $i$ >. For example, given an image and the prompt “Describe <visual\_token\_5>”, we train the model to output a caption for that specific bounding box.

### 3.2 Pretraining Tasks

For multitask pretraining, we formulate seven V+L tasks in a unified text-to-text framework [31]. Table 1 shows example inputs and outputs for each task. Additional material regarding the statistics of the pretraining dataset is shown in Appendix A. Below we give a brief description of each task:

- **Masked Language Modeling (MLM):** Given an image description with  $L$  words, we mask each word  $w_i$  with probability 0.3.<sup>4</sup> The model must learn to reconstruct the original input in the output by predicting the masked words.
- **Image-Text Matching (ITM):** Determine whether a pair of visual-text inputs correspond to each other, predicting true/false after we randomly (with probability 0.5) combine the visual input with either the correct caption or a randomly drawn caption.
- **Captioning:** Produce a textual description of the overall image.
- **Dense Captioning:** Generate a caption for a specified region, denoted using visual sentinel tokens.
- **Visual Grounding:** This can be considered as the dual task of Dense Captioning. Given a description of an image region, the model must predict the visual token for the region that matches this description.

<sup>4</sup>We do not mask words within the task prefix.

- **Visual Question Answering (VQA):** Provide an answer to a query on the given image. Similar to other encoder-decoder architectures, we directly generate candidate answers as opposed to a classification setup.
- **Relationship Prediction:** Generate a sentence describing the relationship between two regions of an image. The sentence is in the format: *Subject Attributes, Subject, Relationship Predicate, Object Attributes, Object*. While subject and object attributes are optional, when available, we include up to two attributes from each attribute category.

### 3.3 Model Pretraining Strategy

During training, we apply teacher forcing and compute the cross-entropy loss between the predicted and target token. As the typical length of the target prediction varies significantly per task, we ensure the losses across tasks are comparable in scale by averaging the loss by the target sequence length first and then again by the number of samples in the batch.

The model is trained with “mixed batches” — where each batch contains examples sampled from any task. Assuming that task  $i$  has  $n_i$  examples, the probability of sampling an example from task  $i$  from all  $j$  tasks is  $p_i = n_i / \sum_j n_j$ . However, as shown in Table 11, the pretraining tasks have a large variance in the number of available examples, which can lead to poor performance on low-resource tasks. Therefore, similar to Raffel et al. [31], we re-adjust the probability  $p_i$  by limiting the maximum number of examples allowed per task. The limit is controlled by a ratio  $R$ , which defines how many more samples are included in training task  $i$  versus the task with the smallest quantity of examples  $n_{min}$ . Therefore,  $\forall j$  tasks, the final probability of sampling an example from task  $i$  becomes  $\bar{p}_i = \min(n_i, R \times n_{min}) / \sum_j \min(n_j, R \times n_{min})$ . In our experiments we set  $R = 3$ .

The model has 6 encoder and 6 decoder Transformer layers, following the architecture of BART-base [20]. As a pretrained object detection model, we selected VinVL [44]. The model was trained on 8 NVIDIA Tesla V100 GPUs for 100K steps, using a batch size of 2048 and the AdamW optimizer with weight decay of 0.01. We apply a linear learning rate schedule with warm-up for 10K steps and a maximum learning rate of  $1e-4$ .

## 4 The EMMA Alexa Skill

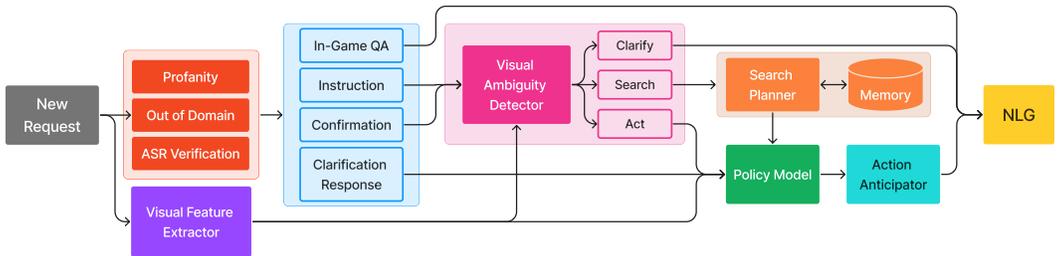


Figure 2. Overview of the EMMA Alexa Skill.

Figure 2 provides a high-level view of the architecture of our Alexa Skill for the Alexa Prize SimBot Challenge. Upon invoking the game, the user is provided with a view of the robot’s perspective in the Alexa Arena [12]. The user can interact with the robot by speaking to their Alexa Echo Show device. The Automatic Speech Recognition (ASR) model from the Alexa Skills Kit (ASK) transcribes the spoken utterance before being passed to our Alexa Skill through the Alexa Arena. Our Skill performs the desired actions and generates responses using Speech Synthesis Markup Language. All responses are conveyed to the user using the ASK’s Text-To-Speech.

**Table 2.** Example dialogue — not from real users — where the mission goal is to defrost a cake with the laser and deliver it to the green desk in the main office. VAD is an abbreviation of Visual Ambiguity Detector. The color coding of triggered pipelines references the skill overview in Figure 2.

Utterance	Triggered Pipeline
<b>User:</b> What is the game about? <b>EMMA:</b> We need to complete the tasks on your screen. Give me step-by-step instructions and I'll do my best to follow them. For more hints - try reading a sticky note! So what's next?	IN-GAME QA
<b>User:</b> Grab. <b>EMMA:</b> Sorry I wasn't able to understand what you need me to pick up. Can you please repeat that for me?	IN-GAME QA
<b>User:</b> Grab the cake from the table <b>EMMA:</b> Done	VAD: ACT ACTION EXECUTION
<b>User:</b> Place it on the red shelf. <b>EMMA:</b> Let's search the robotics lab! <b>EMMA:</b> I see what we are trying to do here. Do you want me to defrost the cake?	VAD: SEARCH SEARCH ACTION EXECUTION ACTION ANTICIPATOR
<b>User:</b> Yes, defrost it. <b>EMMA:</b> Activating the red computer! <b>EMMA:</b> Done, what's next?	CONFIRMATION ACTION EXECUTION
<b>User:</b> Now pick up the cake. <b>EMMA:</b> I remember seeing the cake somewhere here. <b>EMMA:</b> What are we doing with the cake?	VAD: SEARCH SEARCH+MEMORY ACTION EXECUTION
<b>User:</b> Can you go outside? <b>EMMA:</b> Sorry I can't help with that.	OUT-OF-DOMAIN DETECTOR
<b>User:</b> Head to the main office. <b>EMMA:</b> New room - new task. Now what?	ACTION EXECUTION
<b>User:</b> Place the cake on the desk. <b>EMMA:</b> I can see several - which one are you referring to?	VAD: CLARIFY
<b>User:</b> The green one.	ACTION EXECUTION

Since the user is unaware of what EMMA can do, false expectations may jeopardize the interaction and negatively affect the user experience. Therefore, it is critical to filter out-of-domain or even adversarial utterances (Section 4.1) at the start of processing a new request. After validating the user input, we determine how the user would like EMMA to act (Section 4.3). Along with simply performing actions in the Alexa Arena, the best course of action might also include asking the user to disambiguate between objects or asking for confirmation prior to performing some action.

An example dialogue - from not a real user - is illustrated in Table 2, showcasing how the pipelines depicted in Figure 2 are triggered during a mission. Given a user utterance, we may perform multiple consecutive actions before interacting again with the user. During this time, we continually communicate our progress to ensure a smooth interaction. Once we have performed all the actions that we want to do, we hand control back to the user by prompting for another instruction.

#### 4.1 Determining When to Act

From a task-oriented perspective, when the user provides an instruction, they would reasonably expect EMMA to perform an action in the Alexa Arena. However, we observed that not all user queries are directly associated with the game. Therefore, before predicting the action the user would like EMMA to perform, we first validate any user input to determine if we have received an instruction that should not be processed by our pipeline. Examples include adversarial, out-of-domain instructions that are

**Table 3.** User intents which help guide users back to the mission and their responses.

Intent	How we respond to the user
<ask about game>	Briefly explain the UI and how to interact with the Alexa Skill.
<ask about agent>	Prompt the user for instructions, and provide some samples for EMMA to perform.
<greeting>/<admiration>	Greet/Thank the user and prompt them to provide us with an instruction to follow.
<incomplete utterance>	Ask the user to repeat their instruction.

not relevant to the gameplay or users requesting in-game help explaining the purpose of the game or the agent’s capabilities.

#### 4.1.1 Abusive or Out-of-Domain User Input

Each utterance is first filtered through several validators to ensure it is free of any profanity, complete, and in-domain. Since EMMA is a task-oriented skill that must only perform in-domain tasks, we refrain from predicting any actions whenever any of our filters catch the user’s instruction.

**Profanity Filtering** During gameplay, we do not condone the use of profanity or hate speech. Therefore, we use a strict filter to detect any form of profanity or hate speech directed at the Alexa skill.<sup>5</sup> If we detect anything, we return a simple response to the user saying, “*I can’t help with that.*”

**Transcription Verification** For each utterance, the ASR generates token-level confidence scores. After removing any detected wake words, if the average confidence across all words is below 55%, we inform the user that we did not fully understand their request and prompt them to repeat it. As ASR errors are a significant source of user frustration, we do not want to risk the Conversation Experience (CX) by hallucinating the user intent and acting in an undesirable way.<sup>6</sup>

**Out-of-Domain Detection** A crucial aspect of controlling the way EMMA interacts with the environment is by verifying that user input is related to the task at hand. Since EMMA is a generative model, providing an out-of-domain instruction would most likely cause the agent to act in unpredictable ways. Therefore, it is critical to catch out-of-context instructions beforehand.

Nevertheless, it is challenging to exhaustively annotate out-of-domain utterances. To address this, we adapted REDE [16], a few-shot model that has demonstrated strong performance in filtering out-of-domain utterances given a small training dataset. We used a small pool of manually annotated in-domain and out-of-domain utterances. To maintain the high performance of our detector during the interaction phase, we regularly updated the model to account for any failure cases. On our test set, our model achieves an F1 score of 92.30 for out-of-domain detection.

#### 4.1.2 Guiding users and managing expectations

During user testing, we observed that many users requested help during the interaction or were unaware of the mission objectives and the capabilities of EMMA. To address these, we trained a Rasa [8] classifier to detect if the user was asking for help regarding the game or regarding EMMA’s capabilities. In these cases, we prompted the user with an introductory response that clearly stated how to view the goal of the game or what type of actions are executable by our model. Examples of these intents and descriptions of how we responded to users are outlined in Table 3. We also added additional intents to identify greetings, admiration, or when a given user input was incomplete due to hesitation or an error in the transcription.

**Anticipating what users want to do next** We frequently observed users consistently performing the same successive actions in specific environment states. For example, after placing an object on the laser shelf, it is highly likely that the user would like to activate the laser. Therefore, we implemented

<sup>5</sup>We modified the profanity filter from <https://github.com/neorusa/profanity-filter>.

<sup>6</sup>The threshold level was determined after repeated interactions with the Alexa skill.

an Anticipator module to predict the likely immediate next steps given the current environment state. The Anticipator has 18 routines allowing the user to easily operate specific machines within the Alexa Arena. Each routine maps to a sequence of natural language instructions to be executed by the pipeline, using the same action execution process as if the user provided the instruction themselves. If the environment matches all conditions for a given routine, the agent asks the user to confirm the goal. For example, placing any object, such as a mug, on the laser shelf triggers the laser routine and the question “Do you want me to heat the mug?”. If the user answers affirmatively, the agent begins the executing one-by-one a set of instructions that correspond to the routine. Although building the routines for the Anticipator requires domain knowledge about the Alexa Arena, we found that it had a positive impact on the CX and contributed to an increase in the overall user ratings, as it allows players to complete subgoals more quickly.

**Detecting responses to confirmation questions** Our overall design relies on using clarification questions to verify actions inferred based on the user’s instruction. EMMA selectively uses confirmation questions to ask the user to verify action sequences before performing them. Therefore, EMMA must determine whether a response indicates approval or rejection. From observing interactions, asking confirmation questions can help minimize frustration as EMMA is less likely to execute incorrect actions. The confirmation questions are designed to prompt the user to either confirm or deny, however, a user can also respond indirectly to a question or provide a new instruction.

For this component, we trained SetFit [40] for 3-way classification task — outputting one of: *yes*, *no*, or *other*. SetFit uses language representation from a fine-tuned sentence transformer to train a classification head. We used this model because it can be efficiently trained in a few-shot manner. We trained the model end-to-end for 4 epochs on a manually-curated set of 900 utterances, using a batch size of 4 with a learning rate of  $10^{-5}$  on the entire model. The model achieves an F1-score of 98.4% on a test set of 2.6K samples.

## 4.2 Extracting visual features

EMMA relies on object-centric representations derived from an object detector. For our experiments, we chose VinVL [44] because it is trained with a large-scale image dataset that goes beyond the size of COCO; therefore, transferring to other domains can be done in a more robust manner. The VinVL object detector model is a ResNeXt-152 variant that has been pretrained for object and attributes and attribute detection. For our pretraining, we used the publicly-available checkpoint exclusively for object detection. When transferring to the Alexa Arena, we fine-tuned the model with the same data as in Gao et al. [12]. Table 5 shows superior performance of our model compared to the baseline MaskRCNN-ResNet50 model from Gao et al. [12]. Future work will explore additional optimization techniques to further improve the latency of this solution.

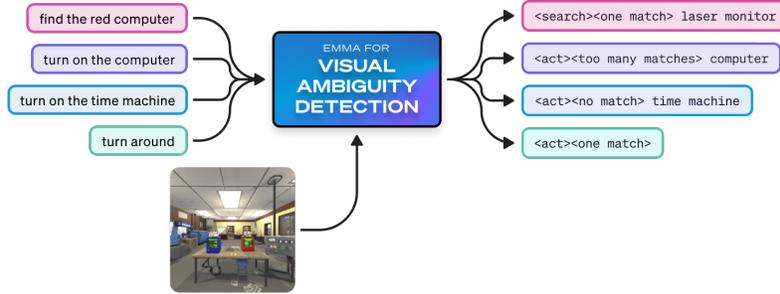
## 4.3 Determining how to act

Once the user input is validated, we process the utterance to determine how the user would like the agent to act. For example, does the user want EMMA to perform an action in the environment or search the room for an object? Additionally, EMMA must determine which is the correct next action. We have decoupled the problem of identifying the type of instruction and predicting an action by using two separate models: *Visual Ambiguity Detector*, and *Policy*.

### 4.3.1 Visual Ambiguity Detection

Given a valid user instruction, the skill can trigger a new search routine, perform an action on the environment, or ask the user a clarification question. The Visual Ambiguity Detection model uses a fine-tuned variant of the EMMA foundation model (Section 3) to determine the best course of action from both the user utterance and the current view of the agent.

The model returns a sequence of tokens that follows a hierarchical intent scheme with three levels. At the bottom level, the Visual Ambiguity Detection model generates whether the user would like the Alexa skill to perform an action or search for an object. Secondly, the model predicts if there is

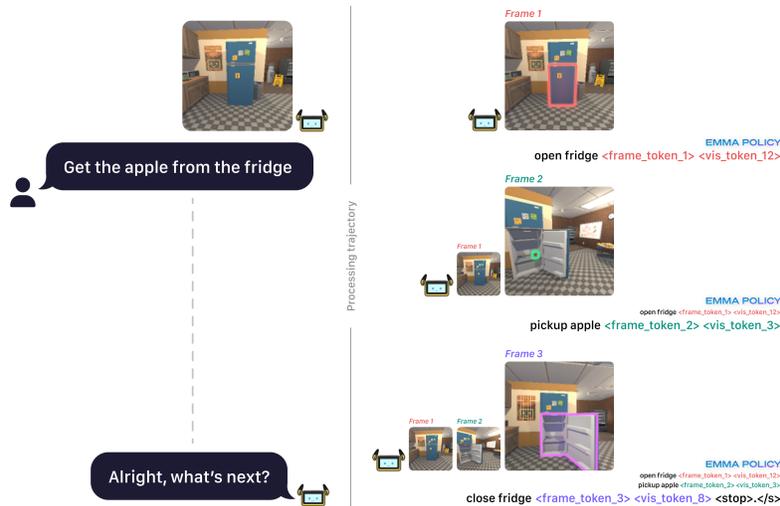


**Figure 3.** Example outputs of the Visual Ambiguity Detection model for the Alexa Arena.

one object present, no objects, or multiple matches of the target object mentioned in the instruction. Finally, it generates the name of the target object.

An example of the output of the model is shown in Figure 3. Given the current observation and the instruction “*Find the red computer*”, the model returns `<search><one match> laser monitor`. However, with the same image, if the instruction is “*Turn on the computer*”, the model outputs `<act><too many matches> computer` since there are multiple valid objects (computers) in the scene. If the instruction targets an object that is not observable, e.g. “*Turn on the time machine*”, the model outputs `<act><no match> time machine`. Finally, for instructions that do not reference any objects, such as “*Turn around*”, the model by convention generates `<act><one match>` with no target object as there is no visual ambiguity.

### 4.3.2 Policy



**Figure 4.** Action Execution: given a user instruction, the model has to predict a sequence of actions. Model predictions are conditioned upon the context experienced thus far.

The Policy model is responsible for executing the instruction from the user. Policy is a multitask model after fine-tuning EMMA on Alexa Arena data. An example output of the Policy model is given in Figure 4. Based on the bottom-level intent of the ambiguity model, Policy performs either 1) visual grounding — matching the textual description of an object to the environment, or 2) action execution — predicting the sequence of actions for the given instruction. We decoupled ambiguity detection and action execution to have more control over our solution. Future work will explore a single massively multi-task model trained to perform visual ambiguity detection, visual grounding, and action execution.

During training, we expose the Policy component to examples from both tasks. Given a search user instruction, the objective is to determine the correct object in the scene that matches the description. Note that since the visual grounding objective is present in the pretraining, we can reuse the same prompts for fine-tuning.

On the other hand, the action execution task is introduced for the Alexa Arena environment. Therefore, we add new task-specific prefixes, for instance: “Act according to the instruction”. These prefixes are concatenated with the user instruction to form the final prompt, e.g., “Act according to the instruction: Get the apple from the fridge.”. We train the model using teacher forcing to predict the sequence of actions for the instruction. Each instruction in the training data can require up to six actions each associated with new observations - frames. In order to differentiate between frames, we use special frame tokens that correspond to their position in the sequence. If a predicted action is an object interaction, the model needs to predict both the correct frame and visual tokens to specify the target object. We use a full stop (“.”) as a delimiter between consecutive actions.

Finally, at inference time, the model operates within a closed loop with the Alexa Arena server, where each action yields an online new observation. We stop predicting new actions whenever the Policy model predicts the <stop> token.

### 4.3.3 Search Pipeline

The search pipeline is triggered every time the Visual Ambiguity Detector predicts that the user intent is to search for an object or to interact with an object that is not in the agent’s view. The agent searches the current room by iterating through selected viewpoints. At each viewpoint, it rotates left by 90 degrees three times. In contrast to the LookAround action, which provides the agent with panoramic views without the user observing any change, this strategy allows the user to observe the game environment and follow the decision-making process of the agent.

Since each room has a maximum of eight viewpoints, the search routine could amount to a large number of repetitive steps. To mitigate this, the agent selects a subset of viewpoints. We assume that the original agent location and each viewpoint can cover an area of radius. This way, we can create a graph where each candidate position is a node and add edges between nodes whose areas overlap. As a result, we turn the viewpoint selection into a Maximum Vertex Coverage problem and apply a greedy algorithm to solve it. We empirically set the radius to four, which leads to selecting up to two viewpoints depending on the room size. After preparing the search plan, the agent starts executing it step-by-step. Each new frame is passed to the Policy model along with the visual grounding prompt and the search instruction. If the object is found, the search terminates; otherwise, the agent continues searching until the plan is exhausted.

We also found that users expect the agent to have some rudimentary form of *visual memory* in order to directly navigate back to objects it has already interacted with. To this end, our system keeps track of the objects encountered in each room, as well as the location from which they were observed. For each step, we obtain the object labels from our vision model along with their 2D coordinates. If any of these objects is not present in the memory, we create a new entry that stores the position of the agent along with the bounding box area. If the object is present in memory, we compare the area stored in the entry with the area of the object from the current view. If the current area exceeds the area written in memory, we override the position and the object area in the memory entity. Since we are not using depth estimation, we used the area of an object as an approximation of how close the object is to the agent at any point. The memory is queried at the beginning of the search routine. If the target object is in the memory for the current room, the agent will navigate to the retrieved location before beginning the search. For a set of special objects that are always located in a known room (for example, the fridge is expected to be in the Break Room), we keep an additional memory of prior knowledge. If the target object is in the prior memory, the agent will start the search by navigating to the known room.

## 4.4 Natural Language Generation

The last step in the agent’s decision-making process is the response generation by the Natural Language Generation (NLG) component. We opted against using a generation model to guarantee

that the response generation process is fully controllable and that the CX is not harmed by model hallucinations. Instead, we developed a rule-based response generator with a large set of template responses inspired by the dynamic dialogue system behind the video game *Left 4 Dead 2* [32].

At the end of each agent turn, the session state is summarized in a dictionary format, and all rules are applied in order to collect the subset of valid candidate response templates. To favor more contextual and novel responses, candidate templates are ordered by the specificity of the corresponding rule and filtered based on which rules have already been used within the session. During the challenge period, we continuously refined the agent responses based on user feedback ending up with 331 templates.

## 5 Dataset

We train our object detector using the provided vision dataset [12]. For the fine-tuned variants of EMMA, we used annotations from three sources: the trajectory data [12], synthetically generated data from the Alexa Arena, and manually-annotated sessions from interactions with the Alexa Arena. The final version of our dataset contains 540K training and 230K validation examples.

**Trajectory data** The trajectory data includes 2,661 task-expert demonstrations for training and 383 for validation, where each task has been annotated by 3 humans. Each task-level annotation corresponds to a sequence of human instructions that describe a sequence of actions. Instructions are additionally accompanied by clarification question-answer pairs. These pairs were used to determine the target of the Visual Ambiguity Detector as described in Section 5.1. We additionally leveraged question-answer pairs for training our Policy, enabling the model to disambiguate given the user input before acting. In cases of ambiguity during inference, we also included the clarification question to the user with their answer as textual input to the Policy model.

**Synthetic instructions** As the trajectory data did not contain examples for certain objects, the performance of our Visual Ambiguity Detector and Policy models was hindered during the early stages of the competition as users explored the environment and performed actions of which the models were unaware. Therefore, we created synthetic instructions to help our Policy model learn the visual grounding objective and expose both models to action-object pairs that are rarely observed or not present in the trajectory data. We continued to refine the set of synthetic instructions throughout the competition, introducing new actions and objects following user interactions.

To create the synthetic dataset, we combined the images with object-level annotations that were used to train the object detector with the object manifest from the Alexa Arena [12] to determine valid action-object pairs (e.g., a fridge can be opened, closed, and can act as a receptacle for other objects). For an action-object pair to be a suitable sample: the area of the ground-truth bounding box must exceed the minimum threshold (Section 5.1), the object state must support the action (e.g., a fridge must be open for the close action to be valid), and the distance between the agent and the object must be smaller than the maximum interaction distance (3m). We limited the maximum number of objects per action to avoid over-saturating the dataset with popular action-object pairs.

Lastly, we created instructions for each synthetic example. We manually created templates for each action type and visual grounding example. In cases of visual ambiguity (e.g., multiple fridges in the scene), we used positional and attribute information from the object-level annotations to refer to the target object. We also manually filtered user sessions to derive actual instructions for these interactions. Next, we used a T5 model [31] to paraphrase each instruction and create multiple instructions for the same action-object pair. With this process, we created a set of instructions that are suitable for every action-object pair.

**Annotating gameplay sessions** During the live interaction phase, we observed that our model may not perform the necessary actions given a user instruction. The primary reasons for the underperformance were incorrect visual ambiguity or policy predictions, out-of-domain errors, or wrong ASR transcriptions. In order to improve our system, we developed a custom app using Gradio [1] to manually-annotate offline gameplay session where our system underperformed. We used our app to review user sessions on a daily basis and debug our system end-to-end. Overall, we created 1300

annotations spanning 164 gameplay sessions. To ensure that the action-object pairs are represented adequately in train and validation sets, we employ a stratified split in terms of object labels. In particular, 80% of the annotations are used for training and 20% for validation. With this setup, we target the performance of the Visual Ambiguity and the Policy model. The remaining components involving profanity, out-of-domain, and confirmation models were updated in fewer annotation cycles.

## 5.1 Data Preparation

**Object Detector** We preprocess images following Gao et al. [12], but with two key distinctions:

1. We categorize Alexa Arena object types into 133 classes (instead of 86). The choice of expanding the class set came from reviewing user sessions during the live interaction phase. We noticed that users were referring to special objects in a particular way that led to clashes with other objects within the same class (e.g, the red/blue shelves and the bookshelves). Using more object classes allowed for more fine-grained object detection that improved model prediction.
2. Consequently, we also updated the minimum area thresholds for each class such that object instances that are visible to users are properly detected by the model. We observed cases where our model was unable to detect objects in the scene even though they were visible to the user, leading to an overall unsatisfying experience. We used the 20% of the smallest object instances per class and then updated certain class thresholds based on manual data inspection. After preprocessing, we obtained 847k images, with 11k images removed during the filtering process.

**Visual Ambiguity Detection** The trajectory data includes only <act> type of instructions, while synthetically generated data includes both instruction types (<act> and <search>). To decide if there are <too many matches> for the target object, we compare its name against the detected object labels. We found it important to apply heuristic filtering to determine if the target object is salient. In particular, an object is considered salient if it is centered or its area is larger than all other instances from the same class. Finally, we need to determine if the instruction is sufficient to disambiguate the target object. For human annotations, we consider an instruction ambiguous if the accompanying clarifications include a “Which” question marked as required. Synthetic instructions with multiple matches are ambiguous by default. We create additional unambiguous instructions using known object properties such as color and material, or the relative location (left or right) of the target object.

In our data, the target object of a given instruction is always in one of the input frames. Hence we need to generate data for the <no match> intent. For human instructions, we keep only the first frame as input and assign the intent <one match> if the target object is located in that frame or <no match> otherwise. For synthetic instructions, we randomly convert <one match> data into <no match> during training. This conversion is necessary because, during inference, we want to know whether or not the agent should first search for an object or act immediately. With a 0.3 probability, we randomly sample an image that does not include any instance of the target object and pass it to the model along with the original instruction.

**Policy** Regarding trajectory data, we decomposed the human instructions so that each training example for action execution corresponds to a single action prediction step. In particular, a training example has 1) a human instruction, 2) a sequence of N previous actions, 3) a sequence of the N+1 observations up to that step, and 4) the follow-up action to be predicted. For the synthetic data, we only applied the same technique as in the ambiguity detector to simulate negative examples for the visual grounding objective. During training with a 0.5 probability, we replace the image of an image-text sample with a randomly sampled image from the dataset, provided that the sampled image does not contain the target object. With this procedure, we create negative examples for the visual grounding objective where the requested object is absent from the image. Finally, we modified the target output of the negative example to match the absence of the object, e.g., if the referenced object is “apple” the target output is “no apple”. Details about the distribution of the data used by both the Policy and Visual Ambiguity Detection components can be found in Figure 7.

## 6 Implementation Details

**Object Detector** We fine-tune the VinVL pretrained checkpoint for 300K steps with a batch size of 4. We set the base learning rate to  $10^{-4}$  and weight decay to  $10^{-5}$  with an SGD optimizer, decaying the learning rate by 0.1 after steps 55K and 75K steps. During training, we use the default preprocessing and image transformations from [14]. The model is trained on 4 RTX 2080 Ti GPUs.

**Fine-tuning EMMA** For the Visual Ambiguity and Policy models, we fine-tune our pretrained model for 20K and 15K steps, respectively, using cross-entropy loss and teacher-forcing. We use a batch size of 512 and the AdamW optimizer with learning rate  $10^{-4}$ , weight decay 0.01, and a linear learning rate schedule with 10% warmup steps.

Data imbalance between tasks is a common pitfall in multitask models, as performance on a high-resource tasks may overtake other low-resource tasks [31]. For example, in the Visual Ambiguity Detection, our dataset is highly-imbalanced towards the `<act><one match>` intent, whereas, for Policy, the `goto` action overpopulates the other action types. To minimize this effect, we employed temperature-scaled sampling.

## 7 Evaluation

In Section 7.1, we report the performance of the pretrained model on image-based downstream tasks to verify the quality of the learned V+L representation. We then evaluate the main models that have been fine-tuned for the Alexa Arena in Section 7.2. In Section 7.3, we report the end-to-end performance of our system on the trajectory data. Finally, in Section 7.4, we show the trend of user ratings during the Alexa Prize SimBot Challenge.

### 7.1 Pretraining Evaluation

**Table 4.** Performance of the pretrained model on downstream image-based tasks. We report the number of pretraining samples as the number of image-text pairs. OFA uses additional vision-only and language-only data. Flamingo uses a significantly larger model and dataset.

Model	# Pretrain Samples	# Params	COCO Captioning				VQA-v2 Accuracy	Ref-COCO Accuracy@0.5	NLVR <sup>2</sup> Accuracy
			BLEU-4	METEOR	CIDEr	SPICE			
VL-T5 [10]	7.6M	172M	34.5	28.7	116.5	21.9	70.3	71.3	<b>73.6</b>
VL-BART [10]	7.6M	172M	25.1	28.7	116.6	21.5	71.3	22.4	70.3
UniTAB [43]	8.1M	211M	36.1	28.6	119.8	21.7	71.0	<b>84.5</b>	—
OFA-base [42]	21.3M	182M	41.0	30.9	138.2	24.2	78.1	82.3	—
Flamingo [2]	> 50B	80B	—	—	138.1	—	82.1	—	—
EMMA	10.2M	113M	<b>36.5</b>	<b>29.7</b>	<b>122.3</b>	<b>22.5</b>	<b>73.2</b>	80.3	70.3

We evaluate the pretrained model on four image-based downstream tasks, which include image captioning, visual question answering, referring expression comprehension, and natural language for visual reasoning. For all tasks, we fine-tune the pretrained model using LM loss for up to 20 epochs. We compare EMMA with strong V+L models with a similar architecture, i.e., single-stream encoder-decoder models. Among models of comparable size, we include OFA-base, which achieves state-of-the-art performance. However, note that OFA-base has 1.6 times the parameters of EMMA and is trained with nearly double the amount of data. For reference, we also add Flamingo [2] — a V+L model which uses more data and has many more parameters (80B) than EMMA (113M).

To evaluate the ability of our model to generate image captions, we use the MS-COCO dataset [23]. We report evaluation results on the Karpathy test split [17] for BLEU-4 [29], METEOR [19], CIDEr [41] and SPICE [3]. For the task of visual question answering, we use the VQA-v2 dataset [13] and report the VQA accuracy [5] on the test-std set. In both tasks, EMMA achieves strong performance on all metrics and is outperformed only by OFA-base and Flamingo.

Referring expression comprehension is a visual grounding task that requires selecting the region described by a given phrase. We evaluate our model’s ability to localize the correct region on the RefCOCOg dataset [26]. A predicted region is correct if the Intersection over Union (IoU) with the ground truth region is larger than 0.5. Our model achieves competitive performance with UniTAB, even though it is almost half the size. Cho et al. [10] hypothesize that the reason for the poor performance of VL-BART is that the use of absolute positional embeddings leads to memorization. However, EMMA also uses absolute positional embeddings and achieves competitive performance.

As a benchmark of our model’s ability to reason over visual inputs, we use the NLVR<sup>2</sup> dataset [39]. The model is provided with a caption paired with two images and has to predict whether the caption is true given the images. VL-T5 achieves the best performance; however, our model is on par with VL-BART, indicating the effectiveness of pretraining with respect to multimodal reasoning.

## 7.2 Alexa Arena Fine-tuning Evaluation

### 7.2.1 Object Detection

**Table 5.** Object detection results for small, medium and large objects. The allowed area of an object in each category is shown in parentheses.

	COCO-mAP			
	All	Small (0-1296)	Medium (1296- 9216)	Large (9216-90000)
Baseline [12]	46.03	37.63	60.41	64.72
Ours	<b>56.70</b>	<b>51.90</b>	<b>89.60</b>	<b>91.90</b>

As shown in Table 5, we evaluate our object detection model using the standard COCO evaluation metric, the Mean Average Precision (mAP), calculated by averaging the precision at IoU thresholds ranging from 0.5 to 0.95 in steps of 0.05. Similar to Gao et al. [12], we set the maximum detection proposals to 100 for evaluation. Importantly, our model is not directly comparable to the baseline model [12], which has a smaller number of target classes, and is trained to perform image segmentation. Nevertheless, our model achieves strong performance with approximately 40% relative improvement for all object sizes.

### 7.2.2 Visual Ambiguity Detection

**Table 6.** Performance for Visual Ambiguity Detection per instruction type. We report the F1 score for intent prediction and the accuracy for the predicted object entity. In addition to performance on our entire validation set, we report separately performance on human instructions denoted by (H).

Intent	F1	Object Accuracy	F1 (H)	Object Accuracy (H)
No Match	0.98	0.98	0.79	0.97
One Match	0.99	0.99	0.98	0.94
Too Many Matches	0.90	0.95	0.73	0.85
All	0.96	0.97	0.83	0.92

Table 6 presents the validation performance of the Visual Ambiguity Detection model. The F1 score evaluates the prediction of the intent, while the Object Accuracy metric evaluates the model’s ability to predict the correct object entity. Overall, the model achieves strong performance on both intent and object prediction. We separate the performance on the subset of human annotations from the trajectory data, which are more varied and unstructured than synthetic instructions. The model retains good performance on “one match”, but finds “no match” and “too many matches” more challenging.

**Table 7.** Results of Policy model for action execution and visual grounding. We report the Exact Match Accuracy, which checks if both the predicted action type and object region are correct, as well as the standalone Action Accuracy and Object Accuracy.

	Action Execution									Visual Grounding
	Total			Unambiguous			Ambiguous			Object
	Exact	Action	Object	Exact	Action	Object	Exact	Action	Object	
EMMA <sub>from scratch</sub>	10.56	99.85	9.85	10.54	99.90	9.86	9.14	99.73	9.33	56.42
EMMA <sub>pretr - clarif</sub>	96.05	99.91	95.72	96.57	99.87	93.19	86.96	97.00	90.07	97.59
EMMA <sub>pretr + clarif</sub>	<b>96.46</b>	<b>99.96</b>	<b>96.13</b>	<b>96.57</b>	<b>99.97</b>	<b>93.19</b>	<b>92.21</b>	<b>99.86</b>	<b>93.91</b>	<b>97.59</b>

**Table 8.** Object accuracy of the Policy model for in-domain and out-of-domain objects. **Table 9.** Accuracy of the Policy model on the test set from GQA-Sim2Real.

Task	In-domain	Out Of Domain	Type	All Objects	Arena Objects
Action Execution	97.36	83.60	Unambiguous	67.21	68.23
Visual Grounding	77.08	60.94	Overall	62.17	62.76

### 7.2.3 Policy

Table 7 shows the results for action execution and visual grounding when comparing the EMMA Policy model against a model trained from scratch and the pretrained model without clarifications. For action execution, we evaluate the model’s ability to predict the correct next action given the instruction and ground-truth action history. For ambiguous instructions, the dialog history includes the instruction followed by a clarification question and answer.

Our results validate the benefits of our pretraining procedure, especially for predicting correct object regions. Predicting the correct action seems relatively straightforward across both models. EMMA trained from scratch has low performance for both EM and OA for all types of instructions, while the pretrained model performs significantly better. With regard to ambiguous instructions, our pretrained model benefits from clarifications. We observe a flat 5.25% performance increase when including the clarification question and user response for predicting the correct action, and a 3.84% performance boost when identifying the correct object.

**Object Generalization Across Tasks** One interesting property that we noticed during the interaction phase was that in some cases, the Policy model could perform cross-task generalization. For example, a model, which was trained to predict actions involving the fridge but had never observed the fridge in visual grounding data, was able to correctly identify the fridge for the unseen task. We suspect that this behavior occurs when an object appears frequently on a task ( $T_1$ ) and is completely absent from the other task ( $T_2$ ), but the model is performing reasonably well on  $T_1$  and  $T_2$ .

To quantify cross-task generalization, we split our synthetic data such that the action execution and the visual grounding tasks do not have object classes in common. We ensured that each task has a representative number of examples and each object class in each task is also appearing frequently. First, we sorted all objects depending on their frequency. Then, we assigned the most frequent object class to the action execution, then the second most frequent object class to the visual grounding, and so on until we included all the object classes. To test the generalization capacity, we created two sets from the validation split of the synthetic data: 1) the *in-domain* split is derived using the method above; 2) the *out-of-domain* split is created by swapping the objects between tasks (e.g, if the fridge was assigned to the action execution, we moved it to the visual grounding). Finally, we fine-tune EMMA for both tasks by monitoring the performance on the in-domain set.

The generalization results are shown in Table 8. Overall, the model exhibits some level of cross-task generalization. For action execution, we notice a performance drop of 14.43% when testing on out-of-domain objects. The performance gap is wider concerning the visual grounding (22.07%), however, the model scores relatively low on in-domain examples. One possible explanation would be

the low number of examples during training as well as the fact that the visual grounding task might require additional training examples to be mastered.



**Figure 5.** Qualitative examples of Sim2Real transfer. The EMMA Policy model fine-tuned for action prediction on the Alexa Arena trajectory data is able to generalize to real images.

**Sim2Real Transfer** We can demonstrate the ability of the Policy model that has been trained on the Alexa Arena action prediction dataset to generalize in the domain of real images. To quantify the model’s ability, we create a synthetic evaluation dataset using scene graphs from the validation set of the GQA dataset [15]. We generate “pick up” and “go to” instructions that refer to objects in the scene along with their attributes. In total, we create 72,870 test samples that include 903 object classes. Out of these, only 56 object classes, such as “table”, “banana”, also appear in the Alexa Arena. We evaluate the accuracy of the predicted objects using an IoU threshold of 0.5. Table 9 shows the results on all object classes, and on the object classes that overlap with Alexa Arena objects. We observe that the performance on all objects does not significantly lag behind the performance on Alexa Arena objects. This shows that the model is able to generalize the action prediction tasks to new objects in the domain of real images. Since the test data are generated, we cannot guarantee that the instructions reference an object without ambiguity when there are multiple objects of the same class within an image. For this reason, we additionally report performance for unambiguous instructions, meaning instructions where there is only one occurrence of the target object class, for which we observe even better performance (see examples in Figure 5).

In the previous section, we show that end-to-end action prediction benefits from pretraining on real images. Here we also show that fine-tuning for action prediction using data from a simulated environment can generalize in the domain of real images without further training. This is an encouraging result for potential future applications reinforcing the value of embodied AI simulation platforms that facilitate research without the cost of experimenting with real robots.

### 7.3 Offline Mission Evaluation

We evaluated the EMMA Alexa Skill on the validation missions of trajectory data [12]. The system is evaluated based on the Mission Success Rate (MSR) and the average Number of Robot Actions (NRA). As this evaluation is offline, we disabled all features which led to asking confirmation or clarification questions that would not receive an answer. Therefore, instead of asking for additional information, EMMA must use the available information and guess the action to take. If EMMA guesses wrong, there is no opportunity for recovery. Without opportunity to self-correct, following

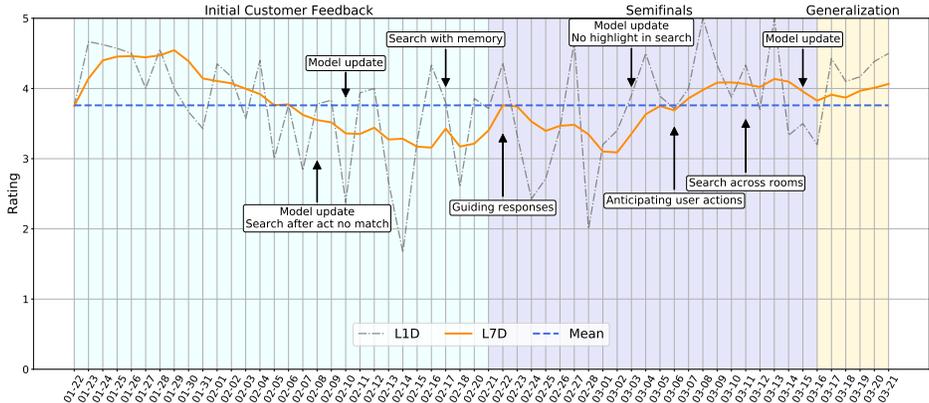
instructions may not be relevant for the current view of the agent, which will likely lead the Policy model to hallucinate actions to perform.

As shown by Table 10, EMMA does not perform as well as other models on this validation set. However, we believe that this difference stems from a discrepancy between the language used in the trajectory dataset versus how users interact with EMMA. This discrepancy arises from the fact that the trajectory data are collected by annotators generating instructions for continuous replays of the agent trajectories while agents experience trajectories step-by-step. For the purposes of the Alexa Prize SimBot Challenge, we have tailored the EMMA Alexa Skill towards minimizing user frustrations and correcting missteps through clarification questions.

**Table 10.** Results comparing the Mission Success Rate (MSR) and the Number of Robot Actions (NRA) taken by EMMA to the models from Gao et al. [12] on the validation trajectory data.

Method	MSR (%)	NRA
Neural-Symbolic	18.19	11.82
VL-model	22.80	12.73
EMMA	16.12	13.03

### 7.4 Human Evaluation



**Figure 6.** Illustration of EMMA’s ratings during the semifinals and generalization phase.

Figure 6 presents the average scores of our system from January 22<sup>nd</sup> 2023 to March 22<sup>nd</sup> 2023, covering part of the customer feedback, the semifinals, and the generalization period. We tried to continuously improve system and add useful features based on user feedback. The main features that have had a positive contribution are the action anticipator, agent memory, and searching across rooms. From the end of the initial customer feedback period to the beginning of the semifinals, we observe a significant fluctuation in our daily ratings. We attribute this fluctuation to technical issues related to the search routine. However, from the second half of the semifinals, there is a clear boost in our ratings — especially during the generalization phase.

## 8 Conclusions & Future Work

We described EMMA, a foundation model for embodied task completion. We designed a careful experimental evaluation to assess many aspects of our model design. First, we used publicly available V+L benchmarks to assess the ability of our foundation model to perform important tasks for an

embodied AI agent such as visual question answering and visual grounding. EMMA is superior (or competitive) to other V+L models of comparable size. Additionally, we demonstrated strong performance on the Alexa Arena dataset in both action prediction and visual ambiguity detection. Finally, our model partially exhibits cross-task generalization and zero-shot transfer from the Alexa Arena to real-world scenes.

We have identified two main areas of improvement for future work. First, during the live interaction phase, it became apparent that guiding the user is key to an enjoyable and successful experience. We found that some users start exploring the game — often without watching the tutorial — and require the agent to help when they get stuck. Our current system can handle certain information-seeking requests and provides feedback for unsuccessful actions. We plan to further improve the ability to help the user by answering user questions about objects. Secondly, we have observed that users may provide high-level commands that our model may not be able to perform. Therefore, we plan to extend EMMA with the ability to execute high-level commands, which we believe will make our system more robust and generalizable.

## References

- [1] Abubakar Abid, Ali Abdalla, Ali Abid, Dawood Khan, Abdulrahman Alfozan, and James Zou. “Gradio: Hassle-free sharing and testing of ml models in the wild”. In: *arXiv preprint arXiv:1906.02569* (2019).
- [2] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. “Flamingo: a Visual Language Model for Few-Shot Learning”. In: *Advances in Neural Information Processing Systems*.
- [3] Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. “SPICE: Semantic Propositional Image Caption Evaluation”. In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, pp. 382–398. ISBN: 978-3-319-46454-1. DOI: 10.1007/978-3-319-46454-1\_24.
- [4] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. “Vision-and-Language Navigation: Interpreting Visually-Grounded Navigation Instructions in Real Environments”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2018, pp. 3674–3683. DOI: 10.1109/CVPR.2018.00387.
- [5] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. “Vqa: Visual question answering”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 2425–2433.
- [6] Dzmitry Bahdanau, Shikhar Murty, Michael Noukhovitch, Thien Huu Nguyen, Harm de Vries, and Aaron Courville. “Systematic Generalization: What Is Required and Can It Be Learned?”. In: *International Conference on Learning Representations*.
- [7] Valts Blukis, Chris Paxton, Dieter Fox, Animesh Garg, and Yoav Artzi. “A Persistent Spatial Semantic Representation for High-level Natural Language Instruction Execution”. In: *5th Annual Conference on Robot Learning*. 2021.
- [8] Tom Bocklisch, Joey Faulkner, Nick Pawlowski, and Alan Nichol. *Rasa: Open Source Language Understanding and Dialogue Management*. 2017. DOI: 10.48550/arXiv.1712.05181. (Visited on 03/18/2023). preprint.
- [9] Emanuele Bugliarello, Ryan Cotterell, Naoaki Okazaki, and Desmond Elliott. “Multimodal Pretraining Unmasked: A Meta-Analysis and a Unified Framework of Vision-and-Language BERTs”. In: *Transactions of the Association for Computational Linguistics* 9 (2021), pp. 978–994. DOI: 10.1162/tac1\_a\_00408.
- [10] Jaemin Cho, Jie Lei, Hao Tan, and Mohit Bansal. “Unifying Vision-and-Language Tasks via Text Generation”. In: *Proceedings of the 38th International Conference on Machine Learning*. International Conference on Machine Learning (ICML). Vol. 139. Proceedings of Machine Learning Research. PLMR, 2021, pp. 1931–1942.
- [11] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G Carbonell, Quoc Le, and Ruslan Salakhutdinov. “Transformer-XL: Attentive Language Models beyond a Fixed-Length Context”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 2978–2988.
- [12] Qiaozhi Gao, Govind Thattai, Xiaofeng Gao, Suhaila Shakhiah, Shreyas Pansare, Vasu Sharma, Gaurav Sukhatme, Hangjie Shi, Bofei Yang, Desheng Zheng, et al. *Alexa Arena: A User-Centric Interactive*

- Platform for Embodied AI*. 2023. DOI: 10.48550/arXiv.2303.01586. (Visited on 03/15/2023). preprint.
- [13] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. “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*. 2017, pp. 6904–6913.
- [14] Xiaotian Han, Jianwei Yang, Houdong Hu, Lei Zhang, Jianfeng Gao, and Pengchuan Zhang. *Image Scene Graph Generation (SGG) Benchmark*. 2021.
- [15] Drew A Hudson and Christopher D Manning. “Gqa: A new dataset for real-world visual reasoning and compositional question answering”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 6700–6709.
- [16] Di Jin, Shuyang Gao, Seokhwan Kim, Yang Liu, and Dilek Hakkani-Tur. “Towards Zero and Few-shot Knowledge-seeking Turn Detection in Task-orientated Dialogue Systems”. In: *Proceedings of the 3rd Workshop on Natural Language Processing for Conversational AI. NLP4ConvAI 2021*. Online: Association for Computational Linguistics, 2021, pp. 281–288.
- [17] Andrej Karpathy and Li Fei-Fei. “Deep visual-semantic alignments for generating image descriptions”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3128–3137.
- [18] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. “Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations”. In: *International Journal of Computer Vision* 123.1 (2017), pp. 32–73. ISSN: 0920-5691, 1573-1405. DOI: 10.1007/s11263-016-0981-7.
- [19] Alon Lavie and Abhaya Agarwal. “METEOR: An automatic metric for MT evaluation with high levels of correlation with human judgments”. In: *Proceedings of the second workshop on statistical machine translation*. 2007, pp. 228–231.
- [20] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. “BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2020, pp. 7871–7880.
- [21] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. “Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation”. In: *International Conference on Machine Learning*. PMLR. 2022, pp. 12888–12900.
- [22] Xiujun Li, Xi Yin, Chunyuan Li, Xiaowei Hu, Pengchuan Zhang, Lei Zhang, Lijuan Wang, Houdong Hu, Li Dong, Furu Wei, Yejin Choi, and Jianfeng Gao. “Oscar: Object-Semantics Aligned Pre-training for Vision-Language Tasks”. In: *European Conference on Computer Vision (ECCV)*. 2020.
- [23] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. “Microsoft coco: Common objects in context”. In: *European conference on computer vision*. Springer. 2014, pp. 740–755.
- [24] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. “ViLBERT: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks”. In: *Neural Information Processing Systems (NeurIPS)*. 2019.
- [25] Arjun Majumdar, Ayush Shrivastava, Stefan Lee, Peter Anderson, Devi Parikh, and Dhruv Batra. “Improving vision-and-language navigation with image-text pairs from the web”. In: *European Conference on Computer Vision*. Springer. 2020, pp. 259–274.
- [26] Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L Yuille, and Kevin Murphy. “Generation and comprehension of unambiguous object descriptions”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 11–20.
- [27] So Yeon Min, Devendra Singh Chaplot, Pradeep Kumar Ravikumar, Yonatan Bisk, and Ruslan Salakhutdinov. “FILM: Following Instructions in Language with Modular Methods”. In: *The Tenth International Conference on Learning Representations, {ICLR} 2022, Virtual Event, April 25-29, 2022*. International Conference on Learning Representations. 2022.
- [28] George Pantazopoulos, Alessandro Suglia, and Arash Eshghi. “Combine to Describe: Evaluating Compositional Generalization in Image Captioning”. In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*. Dublin, Ireland: Association for Computational Linguistics, 2022, pp. 115–131. DOI: 10.18653/v1/2022.acl-srw.11.

- [29] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. “BLEU: a method for automatic evaluation of machine translation”. In: *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics. 2002, pp. 311–318.
- [30] Alexander Pashevich, Cordelia Schmid, and Chen Sun. “Episodic Transformer for Vision-and-Language Navigation”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. International Conference on Computer Vision (ICCV). 2021, pp. 15942–15952.
- [31] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer”. In: *Journal of Machine Learning Research* 21.140 (2020), pp. 1–67. ISSN: 1533-7928.
- [32] Elan Ruskin. “AI-driven Dynamic Dialog through Fuzzy Pattern Matching.” 2012 Game Developers Conference (GDC 2012). 2012.
- [33] Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. “Multitask Prompted Training Enables Zero-Shot Task Generalization”. In: *The Tenth International Conference on Learning Representations*. 2022.
- [34] Rico Sennrich, Barry Haddow, and Alexandra Birch. “Neural Machine Translation of Rare Words with Subword Units”. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2016, pp. 1715–1725.
- [35] Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. “Conceptual Captions: A Cleaned, Hypernymed, Image Alt-text Dataset For Automatic Image Captioning”. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. ACL 2018. Melbourne, Australia: Association for Computational Linguistics, 2018, pp. 2556–2565. DOI: 10.18653/v1/P18-1238.
- [36] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. “ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Seattle, WA, USA: IEEE, 2020. ISBN: 978-1-72817-168-5. DOI: 10.1109/CVPR42600.2020.01075.
- [37] Kunal Pratap Singh, Suvaansh Bhambri, Byeonghwi Kim, Roozbeh Mottaghi, and Jonghyun Choi. “Factorizing perception and policy for interactive instruction following”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 1888–1897.
- [38] Alessandro Suglia, Qiaozi Gao, Jesse Thomason, Govind Thattai, and Gaurav Sukhatme. *Embodied BERT: A Transformer Model for Embodied, Language-guided Visual Task Completion*. 2021. DOI: 10.48550/arXiv.1712.05181. (Visited on 10/20/2021). preprint.
- [39] Alane Suhr, Stephanie Zhou, Ally Zhang, Iris Zhang, Huajun Bai, and Yoav Artzi. “A Corpus for Reasoning about Natural Language Grounded in Photographs”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, 2019, pp. 6418–6428.
- [40] Lewis Tunstall, Nils Reimers, Unso Eun Seo Jo, Luke Bates, Daniel Korat, Moshe Wasserblat, and Oren Pereg. *Efficient Few-Shot Learning Without Prompts*. 2022. DOI: 10.48550/arXiv.2209.11055. (Visited on 03/19/2023). preprint.
- [41] Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. “CIDEr: Consensus-based image description evaluation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 4566–4575.
- [42] Peng Wang, An Yang, Rui Men, Junyang Lin, Shuai Bai, Zhikang Li, Jianxin Ma, Chang Zhou, Jingren Zhou, and Hongxia Yang. “OFA: Unifying architectures, tasks, and modalities through a simple sequence-to-sequence learning framework”. In: *International Conference on Machine Learning*. PMLR. 2022, pp. 23318–23340.
- [43] Zhengyuan Yang, Zhe Gan, Jianfeng Wang, Xiaowei Hu, Faisal Ahmed, Zicheng Liu, Yumao Lu, and Lijuan Wang. “UniTAB: Unifying Text and Box Outputs for Grounded Vision-Language Modeling”. In: *ECCV*. 2022.
- [44] Pengchuan Zhang, Xiujun Li, Xiaowei Hu, Jianwei Yang, Lei Zhang, Lijuan Wang, Yejin Choi, and Jianfeng Gao. “VinVL: Revisiting Visual Representations in Vision-Language Models”. In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Nashville, TN, USA: IEEE, 2021, pp. 5575–5584.

## 9 Acknowledgements

We would like to thank Verena Rieser for her support in the early stages of this project. Additionally, we would like to thank Heriot-Watt University and the Edinburgh Centre for Robotics for supporting our PhD students’ scholarships.

## A Pretraining Dataset

For pretraining, we aggregate five publicly available datasets leading to a total of 10.4M samples. Table 11 shows the pretraining dataset statistics. As multiple datasets source the same underlying images, extra care was taken to ensure that the model was not trained on any instances from the test set.

**Table 11.** Dataset statistics for pretraining.

Dataset	# Images	# Samples	Tasks
VQA-v2 [13]	83K	443K	VQA
GQA [15]	86K	987K	VQA
COCO Captioning [23]	118K	592K	MLM, ITM, Captioning
Conceptual Captions [35]	3M	3M	MLM, ITM, Captioning
Visual Genome [18]	108K	5.4M	MLM, Dense Captioning, Visual Grounding, Relationship Detection
Total	3.3M	10.4M	

## B Policy Dataset

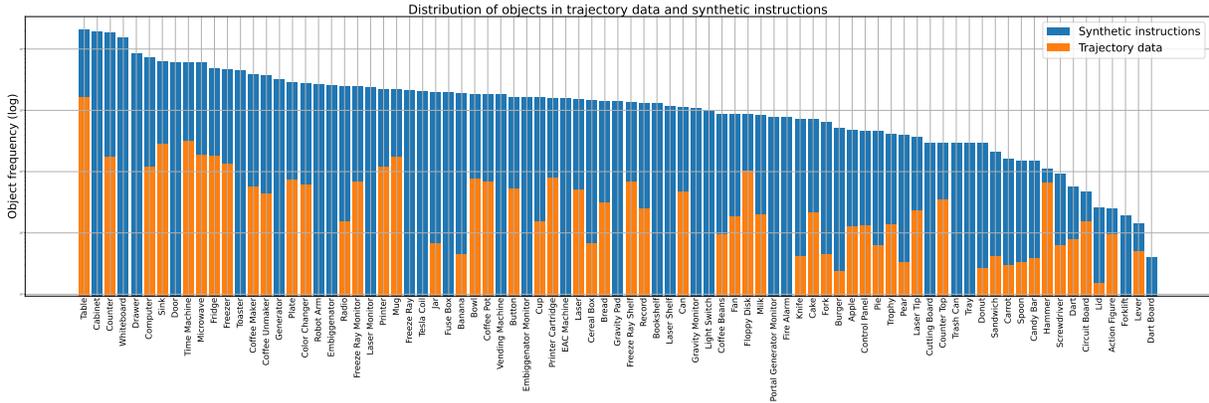
As mentioned in Section 5, we generate synthetic data augmentations to train the Policy model on the tasks of action execution and visual grounding. Table 12 shows the distribution of the synthetic samples per action type. Figure 7 shows the distribution of trajectory and synthetic samples per object. This demonstrates the importance of data augmentations for domain-specific objects, such as the Generator and the Embiggenator, for which the pretraining cannot offer prior knowledge.

**Table 12.** Synthetic examples for the action execution and visual grounding (VG) tasks. Action execution is further decomposed into action types.

Split	Break	Clean	Close	Fill	Goto	Open	Pickup	Place	Pour	Scan	Toggle	VG
Train	11.4K	2.0K	19.6K	7.2K	83.9K	31.8K	53.7K	69.8K	8K	29.7K	28.5K	202K
Validation	2.3K	1K	4.6K	1K	47.3K	15.6K	13.2K	27.3K	2.9K	12.7K	7.1K	102K

## C Natural Language Generation

Section 4.4 summarizes our template-based NLG component. Table 13 shows example rules and templates used to respond when the system receives an `AlreadyHoldingObject` error code. These examples demonstrate how the session state can be used to create from general fallback rules to very specific conditions, allowing the generation of diverse and contextual responses.



**Figure 7.** Distribution of examples for each object. Synthetic instructions are generated because the trajectory data covers a subset of all possible object classes found within the Alexa Arena.

**Table 13.** Examples of the templates used by the NLG component to generate more contextual responses given the current situation.

Description	Received AlreadyHoldingObject error code from the Alexa Arena
Rule	<code>environment_intent_type == "already_holding_object"</code>
Template	<code>&lt;prosody rate="105%"&gt;&lt;amazon:emotion name="disappointed" intensity="medium"&gt; I can't hold anything else - my inventory space is limited &lt;/amazon:emotion&gt;&lt;/prosody&gt;</code>
Description	Received AlreadyHoldingObject error code from the Alexa Arena after trying to pickup something
Rule	<code>environment_intent_type == "already_holding_object" and environment_intent_action_type == "Pickup"</code>
Template	I am sorry - I can only hold one thing at a time
Description	Received AlreadyHoldingObject error code from the Alexa Arena after trying to pickup a known object that has an area $> 200px^2$
Rule	<code>environment_intent_type == "already_holding_object" and environment_intent_action_type == "Pickup" and object_area != null and object_area &gt; 200 and environment_intent_entity != null</code>
Template	I can only hold one thing at a time so I cannot pick up the {environment_intent_entity}.
Description	Received AlreadyHoldingObject error code from the Alexa Arena after the user has directly asked us to pickup something for the second time
Rule	<code>environment_intent_type == "already_holding_object" and intent_type_counter["already_holding_object"] &gt; 2 and current_turn_has_user_utterance</code>
Template	I can only hold on thing at a time - you can see what i'm holding in the bottom right of your screen!