# HouseBot: A Convolutional NLP Framework for a Household Robot

**Zoe Beckman, Naomi Lee, Anessa Petteruti, William Yang**
{zoe_beckman, naomi_lee1, anessa_petteruti, william_yang}@brown.edu

## Abstract

We propose a framework for a household robot HouseBot that performs work done around the household, such as cleaning surfaces, putting away dishes and miscellaneous items, making beds, and turning off electronics at night. We determine which actions HouseBot can perform on a variety of household objects, utilizing a novel framework that combines Facebook's object detection model Detectron2 and a natural language processing model based on ConceptNet. We identify challenges in predicting actionable properties and creating a multi-modal model. This project aims to produce critical insights on how AI can be generalized to act in new environments by identifying objects, properties, and actions.

**Source code:** Code to reproduce these results is available on Github.

## 1 Introduction

Artificial intelligence agents face many challenges when acting in new environments and navigating unknown scenarios. One such challenge is that these agents need to recognize objects – both those they have previously seen before and those they have not – within the new environment. After doing so, the agent must be able to understand the properties of objects as well as actions that can be performed upon each object in order to effectively engage and act in the environment. We can gain insights from the way humans act to build such agents. Humans accomplish these tasks by constantly incorporating visual inputs, symbolic relationships, language processing, and pre-existing knowledge to continuously better their understanding of the world. While computer vision and natural language processing have been studied intensively with the hopes that AI can perform similarly, they have mostly been studied as separate fields and thus are limited in their applications. As such, we seek to use them in tandem, allowing AI agents to navigate unknown scenarios and recognize all types of objects as well as how to interact with them.

HouseBot can be divided into two main components: (1) the Detectron2 object detection model invented by Facebook/Meta and (2) the open-source semantic network ConceptNet. Implemented in PyTorch, Detectron2 was built by Facebook AI Research (FAIR) in 2019 and is a rewrite of Detectron that started with maskrcnn-benchmark (Wu et al. 2019). It includes several models, such as Faster R-CNN, Mask R-CNN, DensePose, Panoptic FPN, Cascade R-CNN, TensorMask, and RetinaNet to aid in region proposal and segmentation tasks. Originating from the Open Mind Common Sense launched in 1999 at MIT Media Lab, ConceptNet helps computers understand the meanings of words and includes knowledge from other crowdsourced resources.

When HouseBot is fed an image of a room scene, it is able to detect which objects are present in the scene and then which actions it could take on the objects in the scene. The list of possible actions is based on the actionable properties available in the AI2-Thor simulator, which assigns a True or False value for whether the robot is able to perform this action on a specificed object. This list of possible

actions includes: toggle, break, fill with liquid, dirty, clean, cook, use up, slice, open, pick up, move, and receptacle.

We evaluated Detectron2's object detection performance when trained on AI2-Thor room scenes using a few metrics discussed in Results. The ground truth data were obtained from the AI2-Thor simulator. For any object in a scene, we queried the relevant field to determine whether this action was possible. In addition to accuracy, we measured both precision (true positives / true positives + false positives) and recall (true positives/ true positives + false negatives).

## 2 Background and Related Work

While much work has been done in isolated subfields of Artificial Intelligence, there is limited research and no integrated framework, to the best of our knowledge, that simulates a household bot by predicting objects in an environment, as well as the actionable properties of the objects.

Much research has been focused on the fields of Natural Language Processing (NLP) and Computer Vision (CV) separately. For instance, there are open-source databases and models such as WordNet, ConceptNet, BERT, and word embeddings. Similarly, researchers continuously seek to improve computer vision techniques and models, such as Recurrent Neural Networks and Convolutional Neural Networks, to interpret images and classify objects. However, a bot that recognizes objects and acts within any environment must understand so much more than simply NLP and CV.

To approach this problem, some have used affordance models in imitation learning and multi-object manipulation tasks, while others have tried learning from demonstrations (Moldovan et al. 2012; Lopes et al. 2007; Niekum et al. 2012). However, this research is often limited in the scope and application of the project, as it is difficult to generalize to more complex objects, actions, and environments. To combat this, some have used goal-based action priors or graph embedding priors to reduce the time needed to make a near-optimal plan (Abel et al 2015; Parikh 2020). This approach, however, requires that the priors be specified by an expert or to be learned from prior experience in related problems, both of which may not be feasible in complex stochastic environments.

As such, we seek to expand upon existing research by creating a framework that generates actions associated with each of the objects in the environment from image inputs of an environment. Given the critical implications on how AI agents act, our model lays the framework for developing intelligent systems that can better understand the objects and environments around them, and in turn more effectively act in such environments.

## 3 Data

### 3.1 AI2Thor Data

AI2-THOR is a collection of open-source enviroments purposed for use in AI applications. It includes iTHOR, a Unity 3D-based simulator of more than 120 rooms (i.e kitchens, living rooms, bedrooms, and bathrooms), objects with visual state changes (e.g. on/off, open/close, hot/cold, and multiple agents in a single scene), RoboTHOR (a simulator of 89 apartments with more than 600 objects), and ManipulaTHOR (an environment in which a robot arm is able to manipulate objects). We used the iTHOR simulator's API to collect images of room scenes on which Detectron2 makes predictions. Simulators provide extremely realistic data for AI applications, which is especially helpful since HouseBot is primarily limited to detecting household objects.

### 3.2 Common Objects in Context (COCO)

Common Objects in Context (COCO) is an object-detection, captioning, and segmentation dataset on which the pre-trained models of Detectron2 are trained. The dataset is large-scale and has many features, including 330,000 images with more than 200,000 labeled, 1.5 million object instances, object segmentation, 80 object categories, 5 captions per image, and 250,000 people with associated

keypoints. We decided to utilize the Detectron2 model, pre-trained on this dataset, in order to make predictions on room scene snapshots taken from the iTHOR simulator. Detectron2 detects objects from 120 iTHOR room scenes, which serve as the predicted labels for the object detection component of HouseBot.

Figure 1: AI2-THOR simulation image of a kitchen floor plan.

### 3.3 iTHOR Room Scenes

We retrieved the ground truth labels for objects found in iTHOR room scenes by examining each simulated scene. iTHOR provides the list of objects in a given scene and with a specified depth value (a larger depth value encapsulates more of the image, and therefore has the potential to detect a greater number of objects than a smaller depth value). A member of the group verified these labels to ensure all predicted objects were found in these scenes. Figure 1 shows an iTHOR room scene with its corresponding ground truth labels and Detectron2-predicted object label list.

**Predicted Object List** = [“refrigerator”, “bowl”, “sink”, “bowl”]
**Ground Truth Object List** = [“fridge”, “bowl”, “sink”, “bowl”]

Figure 2: iTHOR simulation image with ground truth labels and corresponding predicted labels.

iTHOR provides a list of nouns (i.e. objects) that are present in each room scene. Since Detectron2 was designed and trained separately, it predicts a different set of nouns, and the exact words referring to certain objects that Detectron2 predicts can differ. To address this, we considered some terms in the ground truth labels and corresponding Detectron2 datasets to be synonyms. For example, iTHOR provided the object "fridge" – however, Detectron2 is able to detect "refrigerator". For consistency, our ground truth object detection dataset contains only iTHOR objects like "fridge" although our prediction models were performed on both the Detectron2 and ground truth object labels in each image.

iTHOR objects have certain "actionable properties," such as "pickupable," "sliceable," and "openable" as well as "material properties," such as "temperature," "mass," and "salient materials." A list of object types and their corresponding scenes (kitchen, bedroom, bathroom, living room) are shown in ithor.csv in the Github repository, which was scraped from the AI2Thor website. Of importance to this project is the information about which actionable properties are possible to performed on each object, including the properties toggleable, breakable, fillable, dirtyable, useUpable, cookable, sliceable, openable, pickupable, moveable, and being a receptacle. For the Detectron2 nouns in which we cannot access this information from the iTHOR simulator, we assigned ground truth actionable property labels that were consistent with the iTHOR data.

ITHOR Nouns

alarm_clock aluminum_foil apple armchair baseball_bat basketball bathtub bathtub_basin bed blinds book boots bottle bowl box bread butter_knife cabinet candle cd cell_phone chair cloth coffee_machine coffee_table counter_top credit_card cup curtains desk desk_lamp desktop dining_table dish_sponge dog_bed drawer dresser dumbbell egg faucet floor floor_lamp footstool fork fridge garbage_bag garbage_can hand_towel hand_towel_holder house_plant kettle key_chain knife ladle laptop laundry_hamper lettuce light_switch microwave mirror mug newspaper ottoman painting pan paper_towel_roll pen pencil pepper_shaker pillow plate plunger poster pot potato remote_control room_decor safe salt_shaker scrub_brush shelf shelving_unit shower_curtain shower_door shower_glass shower_head side_table sink sink_basin soap_bar soap_bottle sofa spatula spoon spray_bottle statue stool stove_burner stove_knob table_top_decor target_circle teddy_bear television tennis_racket tissue_box toaster toilet toilet_paper toilet_paper_hanger tomato towel towel_holder tv_stand vacuum_cleaner vase watch watering_can window wine_bottle

Detectron Nouns (not including words already present in ITHOR Nouns)

airplane backpack banana baseball_glove bear bench bicycle bird boat broccoli bus cake car carrot cat clock couch cow dog donut elephant fire_hydrant frisbee giraffe hair_drier handbag horse hot_dog keyboard kite motorcycle mouse orange oven parking_meter person pizza potted_plant refrigerator remote sandwich scissors sheep skateboard skis snowboard sports_ball stop_sign suitcase surfboard tie tooth_brush train traffic_light truck tv umbrella wine_glass zebra

Figure 3: List of nouns included in iTHOR and Detectron2.

# 4 Methods

## 4.1 Detectron2

In the object detection algorithm Detectron2, a Residual Network-type neural network introduces residual blocks in order to assist in training. These blocks and skip connections are the heart of ResNet. ResNet is used instead of VGG since it is larger and has more capacity as well as batch normalization.

A Region Proposal Network (RPN) is used to find bounding boxes or anchors, regions of interest for classification. The $\Delta x_{center}, \Delta y_{center}, \Delta width$, and $\Delta height$ are predicted from a baseline reference box. A set of anchors for each of the points in $conv_{width} \times conv_{height}$ is found, and these combinations of sizes and ratios are laid over the original image. Two outputs are defined for

each anchor: the probability that the anchor is an object and the bounding box regression, which adjusts the anchors to fit over the object it is predicting. A binary classification layer outputs the score of being background and the score of being foreground. The bounding box regression outputs $\Delta x_{center}, \Delta y_{center}, \Delta width$, and $\Delta height$ of the anchor. Binary cross entropy loss is minimized for anchors and used as the classification loss. The smooth L1 regression loss is calculated from the foreground minibatch anchors. In post-processing, Non-Maximum Suppression (NMS) is used to get rid of duplicate proposals by replacing proposals with proposals with an Intersection over Union (IoU) value that is greater than a threshold with a higher score proposal. These object proposals have no class assigned to them at this point, the primary reason a feature map is obtained for each proposal. The convolutional feature map is cropped using each proposal and resized to a fixed size $14 \times 14 \times convdepth$ using bilinear interpolation.

A Region-Based Convolutional Neural Network classifies the proposals into one of the classes and a background class and better adjusts the bounding boxes for the proposal according to the predicted class. More specifically, after flattening each proposal feature map, two fully-connected layers, one with N + 1 units where N is the total number of classes and 1 is the background class and the other with 4N units where 4 represents $\Delta x_{center}, \Delta y_{center}, \Delta width$, and $\Delta height$ for each of the N possible classes, are fed into a softmax activation. Proposals with IoU > 0.5 with any ground truth box get assigned to that ground truth label. Proposals with 0.1 < IoU < 0.5 are classified as background, and proposals without any intersection are ignored. Multiclass cross entropy loss is used as the classification loss for all proposals and smooth L1 loss is used as the loss for foreground proposals during training. In post-processing, the class with the highest probability for each proposal is found, and proposals that have a background class as the highest probability are ignored since they are not considered "objects of interest." Objects are grouped by class and sorted by their probabilities.

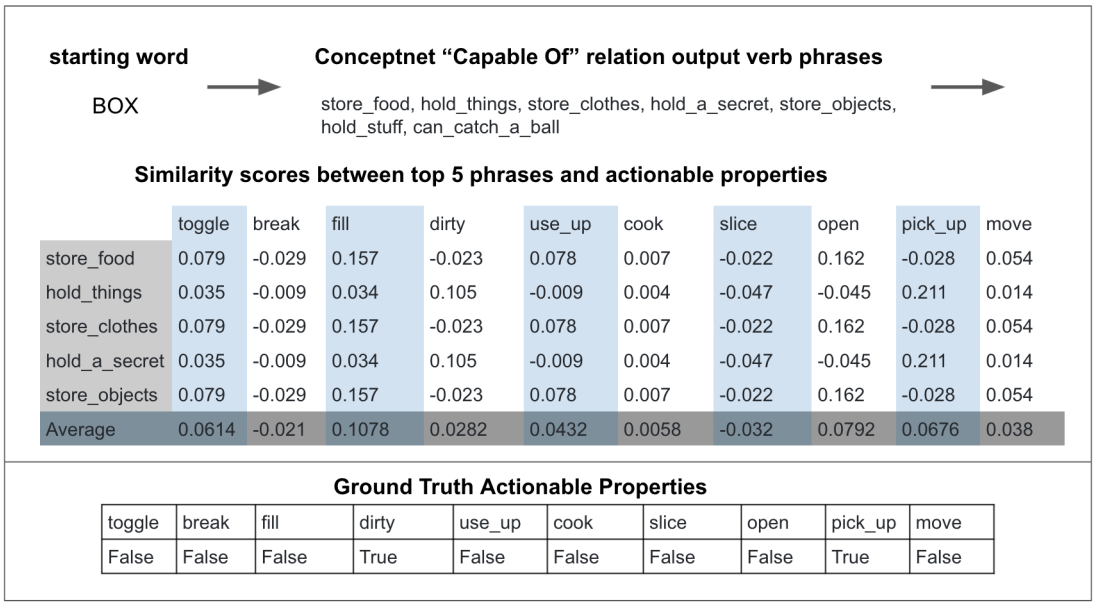
Mean Average Precision (mAP) is used as an evaluation metric in which the model is penalized when it misses a box that should have been detected, detects something that does not exist, or detects the same object multiple times.

### 4.2 ConceptNet

The natural language processing portion of our model utilizes ConceptNet, which provides meaning and relationships between words in a large knowledge base that pulls from a variety of sources, to determine which of the actionable properties would be possible for a certain object. Since there are multiple ways in which these semantic relationships could be harnessed to determine the possible actionable properties, we proposed three main models. First, we created a version ("Threshold Model") that predicts whether it is possible for each property to be performed on the object. We obtain a list of related verb phrases by querying ConceptNet for phrases that fall under a specific relation. For example, querying the "Capable Of" relation for a noun will return a list of phrases that the noun is capable of doing. Since a ConceptNet query for a specific noun may not return any results for a certain relation, we chose to use information from multiple queries to obtain our list of related verbs. We begin by obtaining verb phrases that fall under the "Capable Of" query followed by the "Used For" query if the first query returns no results. These queries return anywhere from zero to more than twenty results and were chosen because they include some semantic information about an object's uses. We then take the first five related verb phrases and take an average of their individual similarities to each of the actionable properties. An example of this process is show in Figure 4. A different ConceptNet query returns the similarity between given words or phrases, although this could also be done manually by measuring cosine similarity between BERT word embeddings of various words/phrases. We then apply a threshold value (for each property) to the average similarity values, in which all values that fall above the threshold indicate that actionable property is predicted as possible (True) to be done and all values that fall below the threshold are predicted as impossible (False).

Second, we created a model ("Max Model") that similarly examines the average similarity between related verb phrases derived from ConceptNet and actionable properties. With this method, we predict one possible actionable property, which is the property with the max similarity score. Since in the ground truth property data, there is a low number of actionable properties that can actually be

performed on each object (an average of 1.56 across all objects), this method aims to predict that possible property, as well as addresses some of the limitations of the Threshold Model discussed below. As in the Threshold Model, we first obtain related verb phrases by querying ConceptNet for verb phrases that fall under the "Capable Of" relation, followed by querying for the "Used For" relation if the first query returns no verbs. Since we are outputting the most likely actionable property but the queries can still return no information which does not yield a max property to choose from, we default to outputting "Pickupable" as the most probable property in which both "Capable Of" and "Used For" relation queries return no results for an object. This property was chosen because it is fairly prevalent as an actionable property in our dataset.




| | toggle | break | fill | dirty | use_up | cook | slice | open | pick_up | move |
|---|---|---|---|---|---|---|---|---|---|---|
| store_food | 0.079 | -0.029 | 0.157 | -0.023 | 0.078 | 0.007 | -0.022 | 0.162 | -0.028 | 0.054 |
| hold_things | 0.035 | -0.009 | 0.034 | 0.105 | -0.009 | 0.004 | -0.047 | -0.045 | 0.211 | 0.014 |
| store_clothes | 0.079 | -0.029 | 0.157 | -0.023 | 0.078 | 0.007 | -0.022 | 0.162 | -0.028 | 0.054 |
| hold_a_secret | 0.035 | -0.009 | 0.034 | 0.105 | -0.009 | 0.004 | -0.047 | -0.045 | 0.211 | 0.014 |
| store_objects | 0.079 | -0.029 | 0.157 | -0.023 | 0.078 | 0.007 | -0.022 | 0.162 | -0.028 | 0.054 |
| Average | 0.0614 | -0.021 | 0.1078 | 0.0282 | 0.0432 | 0.0058 | -0.032 | 0.0792 | 0.0676 | 0.038 |

| toggle | break | fill | dirty | use_up | cook | slice | open | pick_up | move |
|---|---|---|---|---|---|---|---|---|---|
| False | False | False | True | False | False | False | False | True | False |

Figure 4: Diagram showing Threshold Model process and ground truth property labels.

Finally, we proposed a model ("Object Relation Model") in which the actionable properties were known for a subset (20) of objects, like a "knowledge base" containing some objects, and the actionable properties of unseen objects were predicted based on the similarity of the new object to the objects in our knowledge base. With this method, we considered a property to be possible for HouseBot to do, if for the 5 most similar known objects, a majority of the objects could perform that action. However, we realized that this technique faced limitations since HouseBot can only perform on average 1.56 actions on each object, and thus a majority was seldomly met. For instance, given a random knowledge base it was possible that no objects in the knowledge base could have a certain action performed on them, thus always leading to "False" predictions for that property. An improvement on this would weight closer objects that are within a certain distance, but it is also possible that this method would be more reasonable to implement with a larger knowledge base and if objects had more possible actions that could be performed on them.

## 5 Results

We began by analyzing each component of our model separately to fine tune each component.

### 5.1 Detectron2

Detectron2 performs well on detecting objects in images from the COCO dataset and other datasets of real-life scenes. It detected objects in iTHOR scenes with 76.1% accuracy. This accuracy was obtained by determining the number of correctly detected objects in a room scene and the number of objects that were not detected or the number of objects that Detectron2 detected but were

actually not present in the scene. The accuracy of a single image was calculated by determining the fraction $\frac{\text{number of correctly detected objects}}{\text{total number of objects in the predicted object list}}$. These values for all images were averaged together to obtain the 76.1% accuracy.

We stored the predicted objects detected by Detectron2 in a dataset along with the ground truth labels obtained from iTHOR for each image tested on. This information was fed into ConceptNet in order to map possible actions HouseBot could take on the objects themselves.

### 5.2 ConceptNet

We obtained results for both the Threshold Method and Max Method of obtaining possible properties. In the Threshold Method that applies thresholds to the average similarity between each noun's related verbs and the iTHOR properties, we predicted a True/False value for each of 10 actionable properties: Toggleable, Breakable, Fillable, Dirtyable, UseUpable, Cookable, Sliceable, Openable, Pickupable, Moveable. (We combined the Receptacle and Fillable properties into Fillable due to their similarity.) Accuracy was calculated as the number of correctly predicted actionable properties over the total count of actionable properties. When trying to find the optimal thresholds for each property in the Threshold Method, we calculated the average accuracy, precision, and recall for threshold values ranging from -0.1 to 0.25 at .01 increments, with the goal of finding the threshold per property that optimized these values. However, we quickly realized that our model was not sufficiently robust to optimize more than one measure. As shown in Figure 5, for each property, the accuracy increases as the threshold increases, ultimately plateauing around 90%. For each property, the precision starts low, increases slightly as the threshold increases, and then stays at 0 from a certain threshold. The recall begins at 1 and then decreases as the threshold increases, then also plateaus at 0 after a certain threshold. While one property obtains an optimal precision and recall value of around 0.5 for both, most properties do not reach a point where both recall and precision are above 0.3. The reasoning behind this poor performance is discussed in detail in the Discussion.

Using the Max Method, we instead predicted one possible actionable property from the ten using the max average similarity value between the noun's related verbs and iTHOR properties. Accuracy was then calculated as an average across all nouns of whether the predicted property is in fact an actionable property for the ground truth data (True). The accuracy for this section was found when first using "Capable Of" related verbs, followed by "Used For" related verbs if "Capable Of" returned no verbs, followed by defaulting to "Pickupable" as the predicted property if both ConceptNet queries returned nothing (occurred for around 30% of nouns). Testing across all 178 nouns, consisting of all iTHOR nouns and non-overlapping Detectron nouns (lists found in Figure 3) , we found an average accuracy of 34.9%. When only looking at nouns that have at least one true actionable property, this accuracy becomes 41.5%. Interestingly, we found that our ConceptNet model had greater accuracy for the iTHOR nouns than the Detectron nouns when considering these subgroups. We found even greater accuracy when just using the "Capable Of" related verbs and then defaulting to "Pickupable" when no words were returned which occurred for close to half of the nouns (41.1% on all nouns, 46.2% on nouns with at least one actionable property); however, so as to not rely on defaulting to Pickupable, we chose to test our overall pipeline on the "Capable Of" and "Used For" method. Further discussion on the merits of defaulting to a specific actionable property is found in the Discussion.

### 5.3 Overall

For the 119 images that were run through the entire pipeline, the object detection portion showed a 76.1% accuracy in detecting correct objects in the images. This included 262 objects that were identified correctly, which meant Detectron's output either perfectly matched or was a synonym of the ground truth object, 53 objects that were incorrectly identified, as well as 29 extra objects identified by Detectron that were not found in the image. As discussed in the Discussion below, this pipeline leads to repeated predictions of objects, and the Detectron outputs are shown in Figure 6. Using the Max Method to predict one possible actionable property, the natural language processing portion of this pipeline was 37.9%, leading to an overall accuracy, the accuracy in predicting a possible

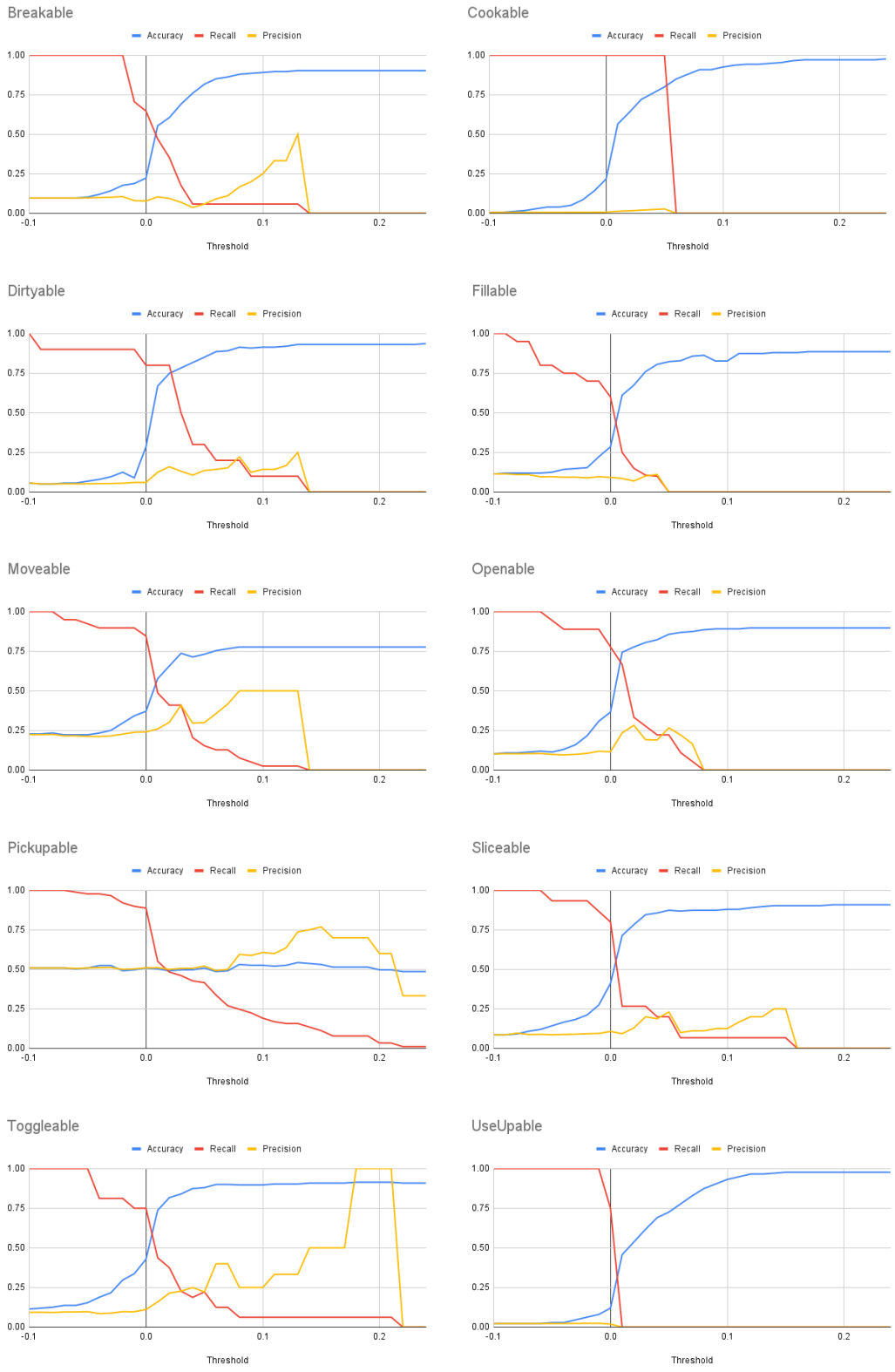



Figure 5: Threshold Model performance ordered by actionable property.

actionable property for an object times the accuracy of identifying the object correctly, of 28.9%. In this result, if Detectron produced a synonym of the correct object in the first step, we ran that noun through the natural language portion and compared the result to the possible actions of the ground truth noun. To situate these results, we also calculated the accuracy when using the ground truth iTHOR nouns in the ConceptNet portion instead of synonyms that were produced by Detectron, which had an accuracy of 36.4% and overall accuracy of 27.7%.

| Word | Count ▼ | Synonym | Count |
|---|---|---|---|
| chair | 51 | | |
| bottle | 27 | | |
| vase | 27 | | |
| sink | 20 | | |
| house plant | 19 | potted plant | 19 |
| sofa | 15 | couch | 15 |
| laptop | 13 | | |
| cup | 12 | wine glass | 2 |
| dining table | 11 | | |
| television | 11 | tv | 11 |
| bowl | 9 | | |
| book | 7 | | |
| fridge | 7 | refrigerator | 7 |
| bed | 6 | | |
| remote control | 6 | remote | 6 |
| alarm clock | 5 | clock | 5 |
| toilet | 5 | | |
| microwave | 4 | | |
| oven | 3 | | |
| cell phone | 2 | | |
| keyboard | 2 | | |
| teddy bear | 1 | | |

Figure 6: Breakdown of words detected in overall pipeline.

# 6 Discussion

## 6.1 Significance of Results

Although at first glance, our Threshold Method appears to lead to good accuracy values, this is simply an effect of having unbalanced ground truth property labels in which about 85% of the actionable properties are False. Therefore, at high thresholds for the properties, we see greater accuracy because the model defaults to predicting False for every property on every object. This realization led us to explore precision and recall as better metrics for our model. However, these results also seem to be limited by the balance of the data. Recall begins very high because it is predicting all true values and thus no properties are missed, although there are many false positives (yielding a very low precision value). As our threshold increases and we predict more False values, the recall drops quickly and precision increases, although neither reaches an optimal value. At the point in which both values become 0, we begin predicting all False for each property, thus causing our true positive value to become 0. Because these results indicate that a simple thresholding model fail where few properties are actually possible, we decided to create our Max Model and use it in the overall pipeline instead.

When considering the accuracy for our Max Method, we took the ground truth values into account. While these accuracies may seem low for predicting just one possible property, each iTHOR noun

only has on average 1.56 actionable properties, so using the max method returns better than chance results (15.6% chance of randoma guessing a correct possible property). It is worth mentioning that we see an increase in accuracy when defaulting to "Pickupable" more (vs. checking for "Used For" verbs too), which may mean that some of our accuracy is due to the large prevalence of Pickupable as an actionable property, although these gains are little when compared to the percentage change in defaulting nouns between the two methods. Therefore, we did a quick experiment to test the accuracy of our ConceptNet model only on nouns that return at least one related verb from ConceptNet (26.6%) and subsequently reduced to those whose ground truth had at least one possible actionable property (31.4%), which highlights both the efficacy of our model when it does not default and that some of our inaccuracy can be attributed to nouns without any possible actionable properties.

When examining the overall results, it is important to note that objects are detected multiple times and in multiple images and that not all possible nouns present in our Detectron dataset will be predicted since we are looking at house scenes. Therefore, the accuracies obtained from ConceptNet are affected by the objects predicted as well as the frequency of each object. This may explain why our accuracy on this part of the model is marginally higher than when we using the full noun dataset when testing our ConceptNet model. We also see that the accuracy when using synonyms appears slightly higher than when using ground truth object nouns. However, when looking directly at these synonyms, we see that of the 7 synonyms detected, ConceptNet is worse at predicting a ground truth possible actionable property for 2 synonyms, the same for 4, and better for 1 vs. when the ground truth noun is run through our Max Method. We can attribute this higher accuracy when using synonyms to the prevalence of "tv" in our images, in which the prediction is better than when using the ground truth value of "television". It is interesting that the predictions are about equally as accurate for synonyms, which is expected since our model relies on similarity between words, but is also hopeful for an existing household robot in which the object detection would most likely be able to return a very large number of words for objects seen.

### 6.2 Challenges

A challenge of Detectron2 was registering a dataset in order to train the object detection model on a custom dataset. We predict that HouseBot's object detection component might have yielded an accuracy greater than 76.1% if we had trained it on a custom dataset of iTHOR room scenes. We were also limited by the nouns that Detectron2 was trained to output, as many were not usually found in household scenes.

Many of the limitations to the ConceptNet and overall models are effects of having an unbalanced dataset. Our initial thresholding model showed limited success due to the small number of possible properties, causing the accuracy to be optimized when always predicting that no actionable properties were possible (all false). Likewise, our third model which relied on object similarity was unable to show success during an initial investigation because it required a large and diverse knowledge base (which hinders both the memory usage and efficiency of this method), so as to represent all actionable properties in high enough frequencies. It is possible that either of these methods would work better for a dataset in which more actionable properties were possible for each noun. It is also worth noting that the ground truth actionable properties are taken from the iTHOR simulator and may not be a perfect mirror of real-life. For example, we noticed that if considering real-life information, many more objects would have been designated as Moveable, Dirtyable, and Breakable.

Our ConceptNet model is also limited by the output from ConceptNet itself. It is possible that for a noun with many possible uses (like a bowl for example), that the average of similarities is not a good method because different uses could be aligned to different properties which could cancel when being averaged, or be missed when only considering a few of the related verbs as we did in our model. Therefore, some system of addressing words with many meanings that could weigh the most common usages could improve this model. Coupled with this, some of the related verb phrases returned by ConceptNet are a bit out of the ordinary since it pulls its knowledge from many sources. For example, one possible "Used For" relation output for an oven is "to get rid of story witches" which is far different from how we normally use ovens for cooking and baking.

Finally, it is important to note that the ConceptNet database is a large database, which could raise some issues when trying to incorporate this knowledge into a standalone system. The downloadable version of ConceptNet, which includes much more information than our models use, required more space than most of our computers had free while conducting this project. We ended up using the Web API to query information from ConceptNet and ultimately stored all possible necessary information in local files since the API sometimes briefly timed out after a few queries, which is not reliable for a model that relies on many queries. Therefore, if a standalone HouseBot were created for public use, it would need to either have the ability to store a lot of data, concisely contain only relevant information, or be reliably connected to an external database to hold all necessary information and allow information to be updated as needed.

## 7 Conclusion

We present HouseBot as a model that recognizes objects and chooses actionable properties in household environments. More specifically, we utilize an accurate objection recognition model, as well as the relationships and associations stored within ConceptNet, to detect objects, choose actions, and most importantly, ensure the generalization across any environment.

Future research can incorporate a graph network that stores and updates the relationships of nouns with other nouns and potential actions, which allows the model to better conceptualize and learn the various associations beyond ConceptNet. Moreover, further research can explore more cohesive and integrated ways to combine the first model that recognizes objects with the second that predicts actionable properties. As our model is one of the first to tackle this intersection of NLP and CV, it is crucial that researchers continue to expand upon the proposed model or construct alternate approaches such that intelligent systems can detect objects and choose actions more appropriately.

## References


Abel, D., Hershkowitz, D. E., Barth-Maron, G., Brawner, S., O'Farrell, K., MacGlashan, J., & Tellex, S. (2015, April). Goal-based action priors. In Twenty-Fifth International Conference on Automated Planning and Scheduling.

Frome, A., Corrado, G., Shlens, J., Bengio, S., Dean, J., Ranzato, M. A., & Mikolov, T. (2013). Devise: A deep visual-semantic embedding model.

Fu, Z., Xiang, T., Kodirov, E., & Gong, S. (2015). Zero-shot object recognition by semantic manifold distance. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2635-2644).

Lopes, M., Melo, F. S., & Montesano, L. (2007, November). Affordance-based imitation learning in robots. In 2007 IEEE/RSJ international conference on intelligent robots and systems (pp. 1015-1021). IEEE.

Kolve, E., Mottaghi, R., Han, W., VanderBilt, E., Weihs, L., Herrasti, A., ... & Farhadi, A. (2017). Ai2-thor: An interactive 3d environment for visual ai. arXiv preprint arXiv:1712.05474.

Moldovan, B., Moreno, P., Van Otterlo, M., Santos-Victor, J., & De Raedt, L. (2012, May). Learning relational affordance models for robots in multi-object manipulation tasks. In 2012 ieee international conference on robotics and automation (pp. 4373-4378). IEEE.

Niekum, S., Osentoski, S., Konidaris, G., & Barto, A. G. (2012, October). Learning and generalization of complex tasks from unstructured demonstrations. In 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 5239-5246). IEEE.

Parikh, N., Horvitz, Z., Srinivasan, N., Shah, A., and Konidaris, G. (2020). Graph Embedding Priors for Multi-task Deep Reinforcement Learning. In The 4th Knowledge Representation and Reasoning Meets Machine Learning Workshop at NeurIPS.

Rosman, B., & Ramamoorthy, S. (2012, November). What good are actions? Accelerating learning using learned action priors. In 2012 IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL) (pp. 1-6). IEEE.

Speer, R., Chin, J., & Havasi, C. (2017, February). Conceptnet 5.5: An open multilingual graph of general knowledge. In Thirty-first AAAI conference on artificial intelligence.

Wu, Y., Kirillov, A., Massa, F., Lo, W., Girshick, R. (2019). Detectron2: https://github.com/facebookresearch/detectron2.