CLIPGraphs: Supplementary Material

Contents

1 Introduction

Humans have always been fond of arranging stuff. Over the course of the development of society, the norms of how we arrange our houses have changed. Still, there are traces of common associations that we as the Human race have managed to stick to.

When a human being is tasked to find us "Headphones" they would intuitively look for it in places such as "Home Office", "Living Room", and "Entertainment Room". This small thought experiment gives us a good idea that subconsciously we have generated some associations between the objects and the rooms they are supposed to be in. Let's dig a bit deeper.

1.1 Traces of Utility in our decision-making process:

For some objects this mapping can be guided by "utility", for example: if asked to find "an apple" in an unknown house, you know it's a perishable food item, and food items are meant to be consumed but also stored appropriately. Thus you would first look for Kitchen, Pantry, or Dining Room (Whatever is available in your house)
More familiar the object, the less the decision-making time to infer the associated room: One way we could further comprehend our inference is by breaking down our thought process into steps. Let's take an example where the object to be found is an unknown name, you don't know if it's a stationery item. a skin care product? or a species of whale? How would you react in such a situation? a logical way to go about this is by asking questions. Your first thought while encountering an object category you don't know is to get to know its utility. Is it a consumable item? Is it cosmetic? Is it somewhat related to electronics? and so on.. Once you are known of its utility, you have an estimate of where different known objects belonging to similar utility are found. This gives you a reasonable amount of confidence in the room you then choose to go search for your unknown object.
This thought experiment establishes 2 facts:

- By successfully answering some of these questions we humans can infer where that object should be found in its correct orientation in *any* house.
- We humans not only have commendable Object-Room mappings in our brains, but we also leverage the object-object relationships developed on the basis of the co-occurrence of these objects, which is again influenced by their common utility.

In this work, we aim to use a graph network to cluster commonly occurring household objects into room categories. The basic intuition behind using a Graph Convolutional Network is to make use of Object-Room Relationships as well as Object-Object Relationships to generate latent representations that are indicative of the categorization of objects into rooms on the basis of their common feature(utility).

Graph Convolutional Networks not only use node features but also use the message passing mechanism between neighboring nodes to generate node representations that take into account the information of neighboring nodes in addition to the node features.

We make use of the Human Preference Dataset [1] to get to know what are some most prevalent human-accepted correct object-room mappings and further use it along with multimodal CLIP [2] features to learn such utility-based clustering using a graph network.

2 Datasets

We used object,room categories and Human annotations for those as provided by Housekeep [1]. Further we curated a visual counterpart to the 268 object categories.

2.1 Human Preference Dataset

This dataset was curated by Housekeep [1] to understand how humans prefer to put everyday household objects in an organized and disorganized house. They ran a study on Amazon MTurk with 372 participants. Here, for a given object-room pair, they asked each participant to classify the available receptacles of that room into 3 categories:

- *misplaced*: subset of receptacles where the object is found in *un-tidy* houses
- *correct*: subset of receptacles where the object is found in *tidy* houses
- *implausible*: subset of receptacles where the object is unlikely to be found either in a *tidy* or *un-tidy* house

They further asked them to rank all the receptacles present in that room. For how they collected this data, filtered it out, and used it for Scene Rearrangement, we guide the interested readers to their paper.

2.2 IRONA Dataset

We created a new dataset by scraping 30 images per object category. These were white background catalog images of the same 268 object categories that Housekeep [1] used to benchmark the Scene-Rearrangement task.

2.2.1 Rooms

We consider the same 17 room categories as used by Housekeep [1]:

Rooms		
bathroom	bedroom	childs_room
closet	corridor	dining_room
exercise_room	garage	home_office
kitchen	living_room	lobby
pantry_room	playroom	storage_room
television_room	utility_room	

2.2.2 Object Categories

The list of 268 object categories we used can be accessed here

3 Knowledge Graph Creation

An important step in learning Object-Room Affinities using our proposed multimodal model was knowledge graph creation using our IRONA Dataset and Housekeep Human Preference Dataset [1]

Figure 1: Representative Image Of Our Web Scraped Dataset ; 1 image per object category

3.1 Nodes

There are currently 2 types of nodes in our knowledge graph

- *Room Nodes:* Since we have 17 room categories, there are 17 such nodes. For our proposed method, the node features for these nodes are generated using the CLIP language encoders.
- *Object Nodes:* For each of the 268 object categories, 15 images are chosen for training, and for each of the 268*15 nodes, we generate node features using the CLIP image encoders[2]. (In our proposed method) However, for baseline comparisons, we also create CLIP Language Embeddings for these 268 Object Categories.

3.2 Edge Weights

For assigning edge weights to various edges between our nodes, we use the ranks provided by Housekeep Human Preference Dataset for each *object-room-receptacle*[1] combination. We further calculate soft scores using algorithm 1 which takes these ranks as input.

[1]"receptacle" was a categorization used by Housekeep [1] to define 128 flat horizontal surfaces in a household where objects can be found - misplaced or correctly placed

For each object-room-receptacle combination, 10 annotators were given just 3 categories to classify the combinations. Therefore for each object-room-receptacle, there could either be a majority of *positive ranks*, *negative ranks*, or *implausible ranks*:

1. For any object-room-receptacle combination if a majority of annotators (> 5) gave positive ranks[2]; we calculate a *positive score*
2. For any object-room-receptacle combination if a majority of annotators (> 5) gave negative ranks[3]; we calculate a *negative score*
3. However, if *implausible* ranks were in majority then we need to modify those a bit. According to Housekeep [1] an *implausible* combination was such a combination that could neither occur in an *untidy* house nor in a*tidy* house. Therefore it accounts for the maximum negative object-room-receptacle affinity, thus we assign -1 i.e. max possible negative score to such object-room-receptacle pairs.

Thus finally we would have a dictionary that records whether the majority opinion for every object-room-receptacle was *positive*, *negative* or *implausible*(-1)

Algorithm 1 Generating ORR Positive Soft Scores

```
1:  for each object in objects do
2:      for each room in rooms do
3:          $ranks \leftarrow object - room - receptacle - ranks$
4:          $score\_dict \leftarrow \{\}$
5:          for each receptacle in the room do
6:              $combination\_score \leftarrow []$
7:              for each rank given for the combination do
8:                  if $rank > 0$ then
9:                      $combination\_score.append(1/rank)$
10:                 end if
11:             end for
12:             if $len(combination\_score) >= 5$ then
13:                 $score\_dict[combination] = sum(combination\_score)/max\_len\_pos$[4]
14:             else
15:                 $score\_dict[combination] = 0$
16:             end if
17:         end for
18:     end for
19: end for
```

3.2.1 Correct Object-Room Mappings

To get the correct object-room mappings, we compare for a given object which object-room-receptacle has the highest *positive* score. We assign that room as the correct object-room mapping.[5]

[2]A receptacle with "+1" rank is a more appropriate for an object as compared to a receptacle with a "+2" rank for the same object-room pair.

[3]A receptacle with "-1" rank is a receptacle where humans are more prone to keep objects in an *untidy* state of the house as compared to a receptacle with a "-2" rank for the same object-room pair.

[4]For a given object, max_len_pos/neg/imp is the maximum number of annotators that gave positive/negative/implausible ranks across all room-receptacle pairs.

[5]Our future extension would be to extend this mapping to top-K correct rooms

Algorithm 2 Generating ORR Negative Soft Scores

```
for each object in objects do
    for each room in rooms do
        $ranks \leftarrow object - room - receptacle - ranks$
        $score\_dict \leftarrow \{\}$
        for each receptacle in the room do
            $combination\_score \leftarrow []$
            $min\_neg\_rank = min(all\ ranks\ for\ the\ ORR\_combination)$
            for each rank given for the combination do
                if $rank < 0$ then
                    $rank = rank + min\_neg\_rank + 1$
                    $combination\_score.append(1/rank)$
                end if
            end for
            if $len(combination\_score) >= 5$ then
                $score\_dict[combination] = sum(combination\_score)/max\_len\_neg$[4]
            else
                $score\_dict[combination] = 0$
            end if
        end for
    end for
end for
```

Algorithm 3 Generating ORR Implausible Soft Scores

```
for each object in objects do
    for each room in rooms do
        $ranks \leftarrow object - room - receptacle - ranks$
        $score\_dict \leftarrow \{\}$
        for each receptacle in the room do
            $combination\_score \leftarrow []$
            for each rank given for the combination do
                if $rank == 0$ then
                    $combination\_score.append(-1)$
                end if
            end for
            if $len(combination\_score) >= 5$ then
                $score\_dict[combination] = sum(combination\_score)/max\_len\_imp$[4]
            else
                $score\_dict[combination] = 0$
            end if
        end for
    end for
end for
```

3.2.2 Incorrect Object-Room Mappings

Once we've created the correct object-room GT mapping, it still leaves the other 16 rooms' edge weights to be allotted. So for each such object-room pair; there can be 3 cases:

1. **All/majority positive receptacles**:we assign $-\epsilon$ as edge weights
2. **All/majority negative receptacles**: we assign the mean of all the negative scored receptacles
3. **All/majority implausible receptacles**: we assign the mean of all the negative scored receptacles. (since we had already assigned implausible receptacles with -1)

For example, for a particular object-room-receptacle combination, the calculation of positive soft scores is shown:

$$\begin{aligned}
\text{knife-bottom\_cabinet ranks:} &\quad -1, -3, 0, 1, 3, 2, -2, 5, -1, 4 \\
\text{Filter the negative ranks:} &\quad 1, 3, 2, 5, 4 \\
\text{Take the reciprocal of ranks:} &\quad \frac{1}{1} + \frac{1}{3} + \frac{1}{2} + \frac{1}{5} + \frac{1}{4} = 2.367 \\
\text{Calculate the mean of reciprocal ranks:} &\quad 2.283/5 = 0.45
\end{aligned}$$

For each object, the ground-truth room is decided by choosing the room containing the highest-positively scored receptacle for that object. Every other room in the domain is assigned the mean negative soft score(for a given object-room pair) of all the receptacles present in that room.

3.3 Overall

Table 1 summarizes various information about our knowledge graph that is being used for training purposes.

Statistics About our Knowledge Graph	**Value**
#Nodes	4020 Object Images Nodes & 17 Room Nodes
#Edges	7,66,649
Self Loops	Yes
Train Images	268*15 Images
Test Images	268*10 Images
Val Images	268*5 Images
Types of edges	weighted undirected

Table 1: Statistics for the Knowledge Graph created using the web scraped dataset

3.4 Comments on node features

For our proposed method we used 3 architectures of CLIP as given by OpenCLIP [3]: ViT-H/14, and RN50 have dimensionality of 1024, and ConvNeXt-base has a dimensionality of 512 features. These were chosen taking into consideration the datasets they were trained on and their performance in other embodied AI tasks.

4 Loss Function Ablations

We considered two performance measures:

1. **mAP:** The mean average precision (mAP) is the average of precision scores at different recall values for each instance of an object category, and the mean over all the object categories. For a given object, Average Precision is calculated by:

$$AP = \sum_n (R_n - R_{n-1}) P_n$$

Where, P_n and R_n are Precision and Recall values at the n^{th} threshold. Taking a mean over all the objects, gives us the final mean Average Precision.

2. **Top k Hit Ratio:** The average fraction of object categories for which the ground truth correct room was among the Top k estimates from our framework.:

$$\text{Top-k HR} = \frac{1}{|O|} \sum_{o \in O} \mathbb{1}(R_o \cap T_o \neq \emptyset)$$

where O is the set of objects, R_o is the set of top-k rooms recommended for object o, T_o is the ground truth room for object o, and $\mathbb{1}$ is the indicator function that returns 1 if the condition is true and 0 otherwise.

4.1 vs existing loss functions

Loss Fn	mAP
Margin[4]	0.371
Triplet[5]	0.51
Ours	**0.85**

4.2 Hyperparameters

We chose a modified version of the loss function used by [6]. The effectiveness of our loss function depended on our sampling technique. We tuned

1. Batch Size
2. Number of negative nodes sampled per batch for each epoch.

4.2.1 Batch Size

We experiment on the number of batches of anchor, positive and negative nodes sampled per epoch for loss computation. As evident from the figure, we achieve the best performance at batch size = 15. Thus we continue our experiments with this value. The comparison is shown in Figure 3

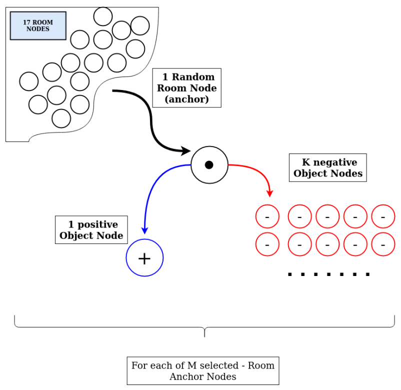

17 ROOM NODES
1 Random Room Node (anchor)
K negative Object Nodes
1 positive Object Node
For each of M selected - Room Anchor Nodes

Figure 2

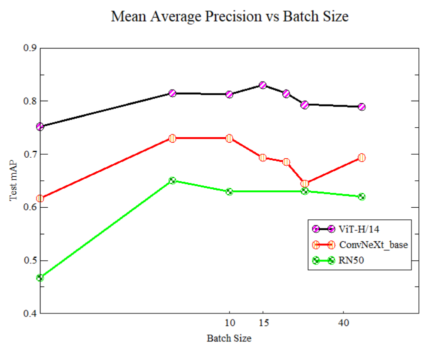


Figure 3: Variation of testing metrics with number of batches used for contrastive loss

4.2.2 Number of negatives

We compare the performance of our model by changing the number of negative nodes sampled per anchor point for computing the contrastive loss. The results are compiled in figure 4. As evident from the figure, the best performance was observed at negative samples = 40. Thus we fix the total number of negatives per anchor as 40.

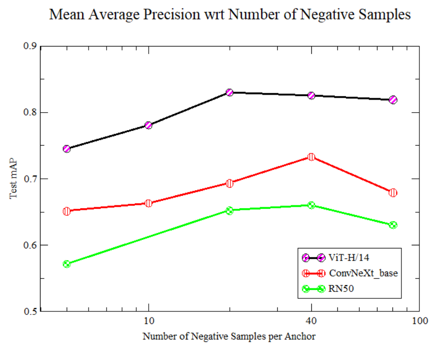


Figure 4: Variation of testing metrics with different numbers of negative samples used per anchor.

5 Our Model: GCN

Table 2 shows the hyperparameters used for the model.

#	**Hyperparameter**	**Value**
1	Node Feature Size	512(ConvNeXt) / 1024 (Others)
2	Output Node Embedding	128
3	GCN[7] Layers	3
4	Learning Rate	10^{-3}
5	Learning Rate Schedule	StepLR: step size=1000 , $\gamma = 0.25$
6	Temperature	0.01
7	Batch Size [Loss]	15
8	Negatives Per Batch	40
9	Epochs	5k

Table 2: Hyperparameter choices for our Graph Based Network to learn latent representations of CLIP Visual Encoder Features

6 Baseline Predictions

6.1 GPT-3

To obtain baseline results, we query GPT-3 to rank the 17 rooms for each object like this:

Which of the following rooms would you expect to find a **knife block** in? Please rank in decreasing order of likelihood: bathroom, bedroom, child room, closet, corridor, dining room, exercise room, garage, home office, kitchen, living room, lobby, pantry room, playroom, storage room, television room, utility room.

We use such queries to obtain room rankings for each of 268 objects and use that to obtain baseline mAP and hit ratio for GPT-3, and the results are shown in Table 3

	Test mAP $\Uparrow$	Hit-Ratio $\Uparrow$		
		Top-1	Top-3	Top-5
GPT-3	0.66	0.52	0.76	0.81

Table 3: Results for object-room mappings based on queries to GPT-3

The room rankings for each object category by querying GPT-3 is compiled in the file: gpt_pred.txt

6.2 Language Encoders

The code for generating the statistic metrics as well as the object-room mappings corresponding the each language baseline is available at CLIPGraphs GitHub Repository.

6.3 Our Predictions

Similar rankings for every object category based on our model can be generated by running the script available at: CLIPGraphs GitHub Repository

7 Qualitative Results

Our model was trained on clean white background images to predict the most appropriate room. To test its performance on real-world images we ran a few runs on a mobile phone by clicking a photograph and feeding it to our model. We fed our model images of objects out of the training set to see its generalization capabilities, we report some success and failure cases for the same.

7.1 Success Cases

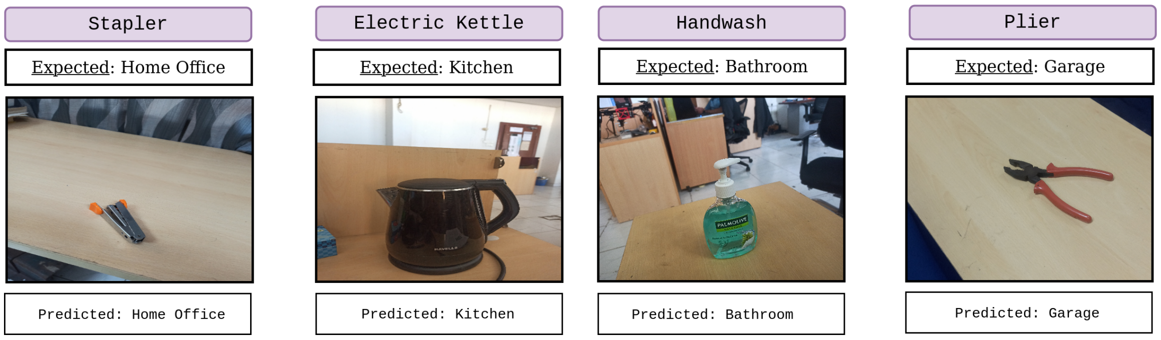
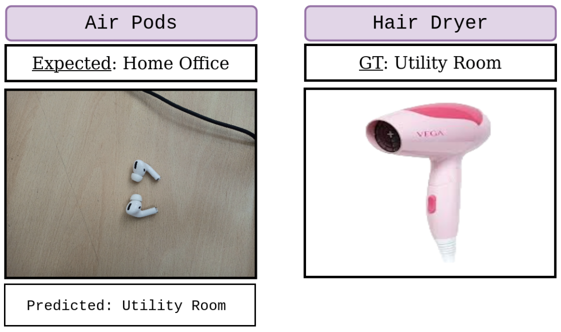


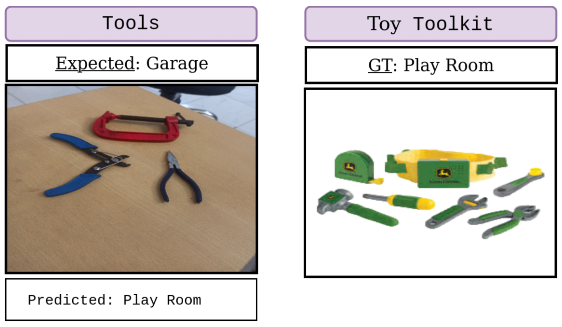
Figure 5: Success of the model on unseen object category images (absent in the training set)[6]

7.2 Failure Cases

Figure 6 shows some specific limitations of our models. In Figure 6a when we input the image of earpods (not a part of the training set) to the model, the top output prediction is utility room. The model confuses the images of earpods to similar image of hair dryer, since it does not contain knowledge of scale. Figure 6b shows the limitation of the model in identifying real objects with similar category of toys.



(a) Failure to determine correct room for object category *earpods* (not in our train set) because it was structurally similar to *hair dryer* category that was in our training set



(b) Failure with composite object categories; *tools* was not a category in our training set, but they were incorrectly associated with the *play room* because they were structurally similar to the *toy toolkit* that was in the training set.

Figure 6: Failure cases of our model

[6]Since these categories were unseen thus we didn't have any ground truth available, thus we mention the *expected* room on the basis of our commonsense

Apart from testing our model on unseen categories, we also try our model's generalization capabilities on real-world noisy images. For this experiment, we generated 4 different scenarios. The representative results for 3 object categories are shown below:

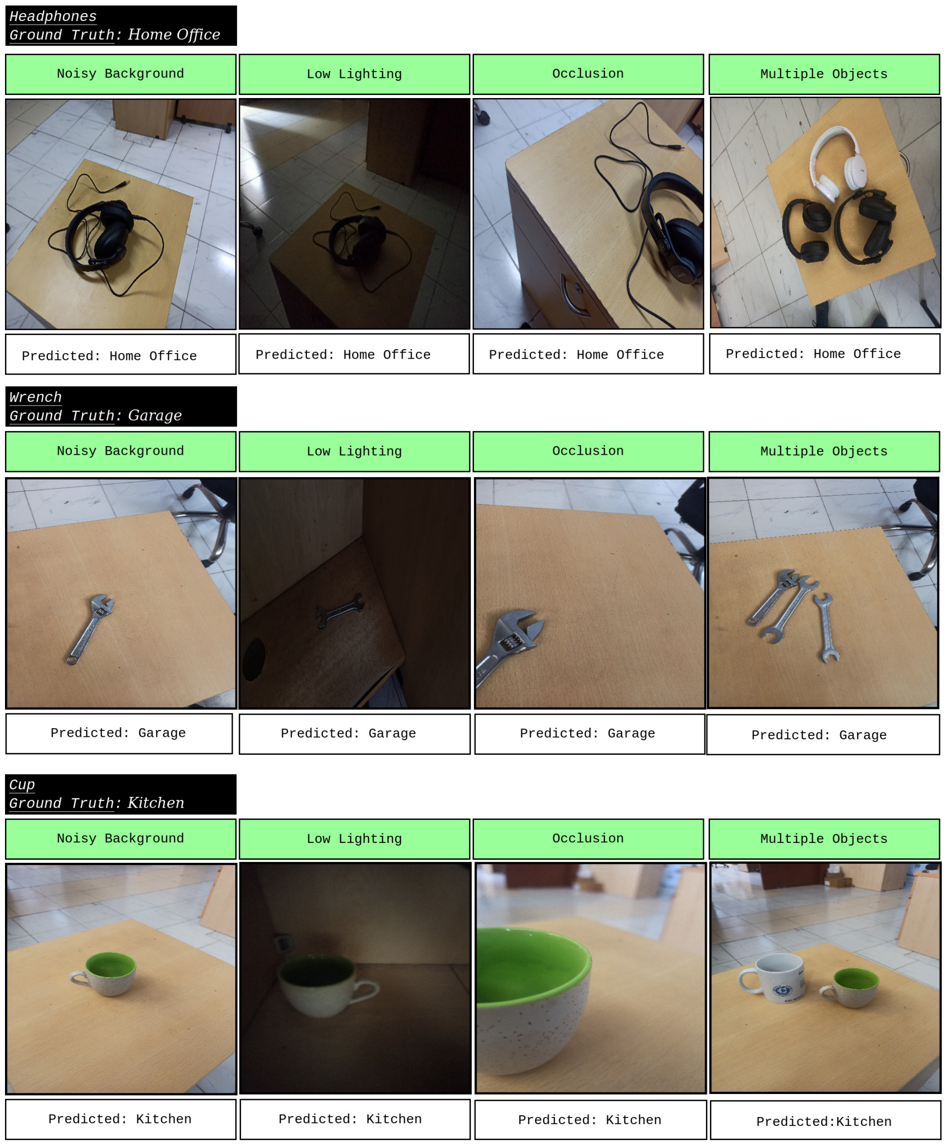


Figure 7: Qualitative result of using our framework with images of previously *seen* objects but in noisy backgrounds. In each case, the object's room association was estimated correctly demonstrating broad applicability of our method

8 Some additional Plots

8.1 t-SNE

Using t-SNE to visualize the high-dimensional embeddings, we observe initial random nodes in Figure 8, where each color represents objects of a unique room. As the model trains, we observe clustering in the embedding space to cluster objects belonging to the same room [Figure 9]

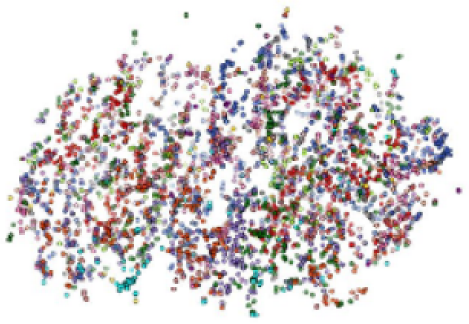


Figure 8: Untrained TSNE

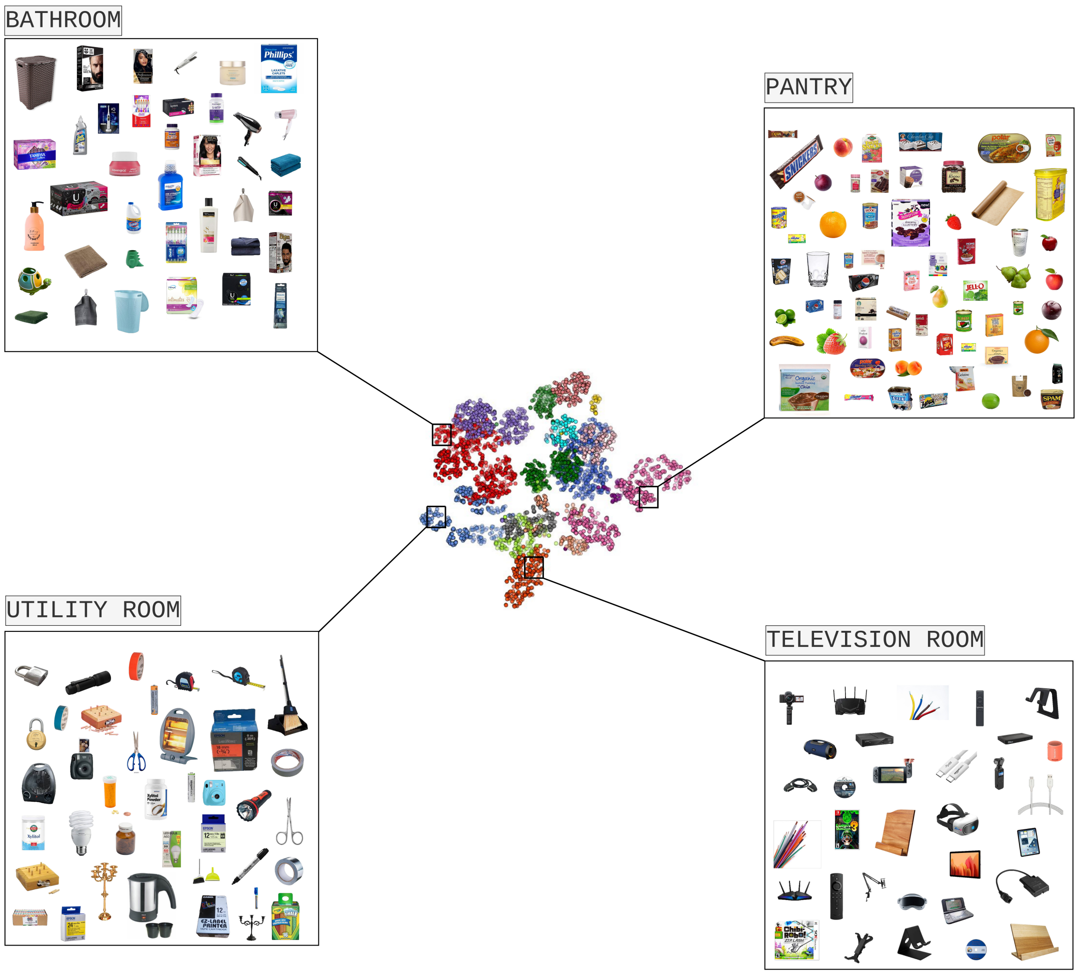


Figure 9: t-SNE visualization of our embeddings on the test split of the Web Scraped Dataset. The boxes show images of objects belonging to the same rooms getting clustered [a]

[a]For a more interactive view of this figure, check out our website: https://clipgraphs.github.io

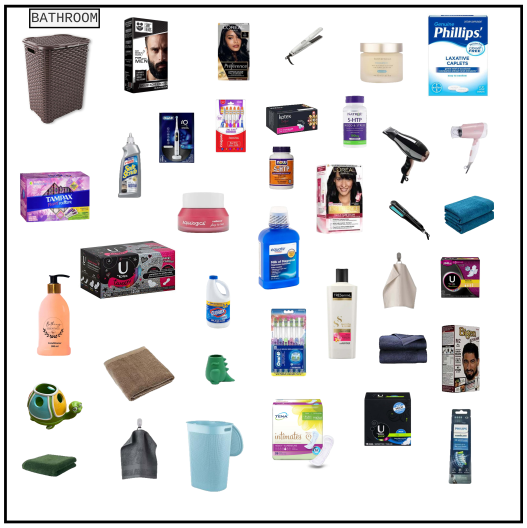


Figure 10: Image showing objects that got clustered in the t-SNE corresponding to ”Bathroom” room category

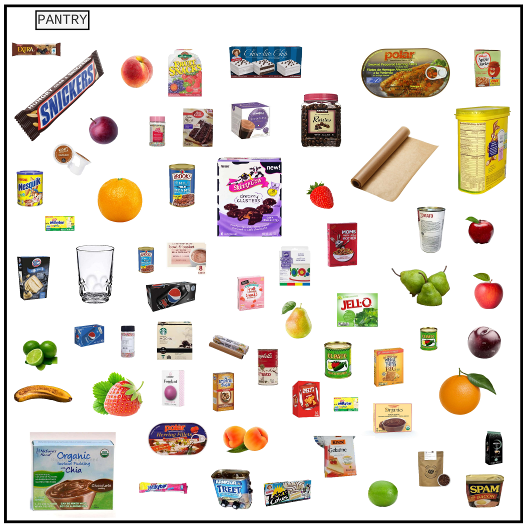


Figure 11: Image showing objects that got clustered in the t-SNE corresponding to ”Pantry” room category

References

[1] Y. Kant, A. Ramachandran, S. Yenamandra, I. Gilitschenski, D. Batra, A. Szot, and H. Agrawal, “Housekeep: Tidying virtual households using commonsense reasoning,” in *European Conference on Computer Vision*, 2022.

[2] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, “Learning transferable visual models from natural language supervision,” in *International Conference on Machine Learning*, 2021.

[3] M. Cherti, R. Beaumont, R. Wightman, M. Wortsman, G. Ilharco, C. Gordon, C. Schuhmann, L. Schmidt, and J. Jitsev, “Reproducible scaling laws for contrastive language-image learning,” *ArXiv*, vol. abs/2212.07143, 2022.

[4] F. Schroff, D. Kalenichenko, and J. Philbin, “Facenet: A unified embedding for face recognition and clustering,” *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 815–823, 2015.

[5] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, and S. Levine, “Time-contrastive networks: Self-supervised learning from video,” *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1134–1141, 2017.

[6] N. M. M. Shafiullah, C. Paxton, L. Pinto, S. Chintala, and A. D. Szlam, “Clip-fields: Weakly supervised semantic fields for robotic memory,” *ArXiv*, vol. abs/2210.05663, 2022.

[7] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in *International Conference on Learning Representations (ICLR)*, 2017.