CLIPGraphs: Supplementary Material

April 4, 2023

Contents

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.

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 utility.

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

This categorization has its shortcomings as well. Humans not only arrange household items on the basis of utility. Sometimes these mappings can be guided by "ease of use" for example: even though "utility" wise "water bottle" is supposed to be in and around the "kitchen", through experience, we know that we are more likely to find it in places where Humans are more prone to spend time. Object retreival in such homes is quite tricky just on the basis of generalised uility based categorization of objects.

2 Datasets

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

2.1 Human Preference Dataset

This dataset was curated by Housekeep [**?**] to understand how humans prefer to put everyday household objects in an organized and disorganized house. They ran a study on Amazon MTurk [**?**]. 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

- correct: 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 object categories were kept same as that of the object categories that Housekeep [**?**] used.

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

2.3 Rooms

Following Housekeep, we consider the same 17 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	

3 Knowledge Graph Creation

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. 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 the node features are generated using the CLIP image encoders.

3.2 Edge Weights

Using ranks provided by Housekeep for object-room-receptacle combinations, we calculate soft scores using the algorithm 1

Algorithm 1 Pseudocode for getting object-room-receptacle soft scores

```
for each object in objects do
    for each room in rooms do
        ranks = object-room-receptacle-ranks ▷ ranks is a dictionary of ranks for each combination of object and receptacle for the room (from housekeep)
        score_list = []
        for each combination of object and receptacle do
            combination_score = 0
            for each rank given for the combination do
                if rank>0 then
                    combination_score += 1/rank
                end if
            end for
            score_list.append(combination_score)
        end for
        find the maximum length of all score lists
        for each combination of object and receptacle do
            if len(score_list)>5 then
                calculate the average score
            else
                set the score to 0
            end if
        end for
    end for
end for
```

Here, we filter out the negative ranks and get a positive score for each object-receptacle combination.

We repeat the same algorithm for the negative ranks, to obtain negative scores for each object-room-receptacle pair.

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 of receptacles in that room.

3.3 Overall

Table 3 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

4 Loss Function Ablations

4.1 vs existing loss functions

4.2 Hyperparameters

An important design choice for the effective model learning is how we sample the positives,negatives and the anchor nodes.

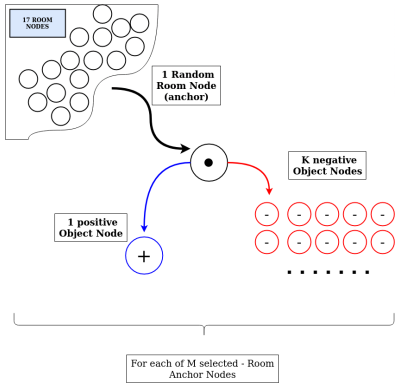


Figure 2

4.2.1 Batch Size

We experiment on the number of batches of anchor, positive and negative nodes sampled per epoch for loss computation, and fix the batch size at 15. The comparison is shown in Figure 3

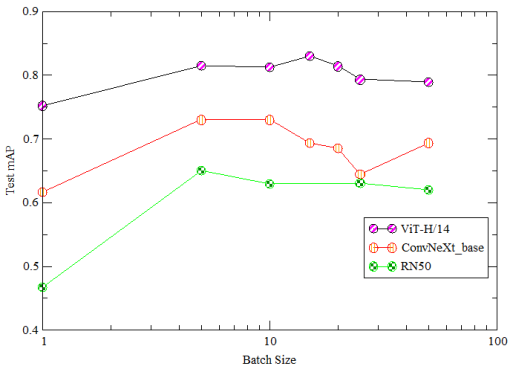


Figure 3: Variation of testing metrics with number of batches used for contrastive loss

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, and we fix the total number of negatives per anchor as 40.

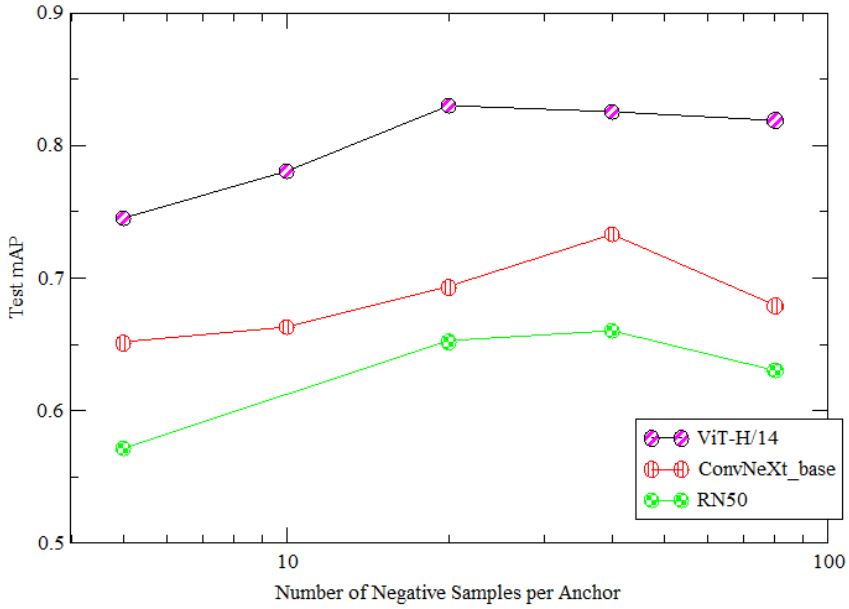


Figure 4: Variation of testing metrics with different number of negative samples used per anchor.

5 Our Model: GCN

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?
Rank in decreasing order of likelihood: 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.
Please give list separating objectname with colon from room names, and comma separated room
    names
```

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 2

	Test mAP ⇑	Hit-Ratio ⇑		
		Top-1	Top-3	Top-5
GPT-3	0.66	0.52	0.76	0.81

Table 2: 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

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 at these settings:

Details	Value
Device	Redmi Note 8 Pro
f	*f/1.89*
Flash	No

Table 3: Details of Real World Images

clicked a photograph and fed it to our model. We tested this in 4 different scenarios. The representative results for 3 object categories are shown below:

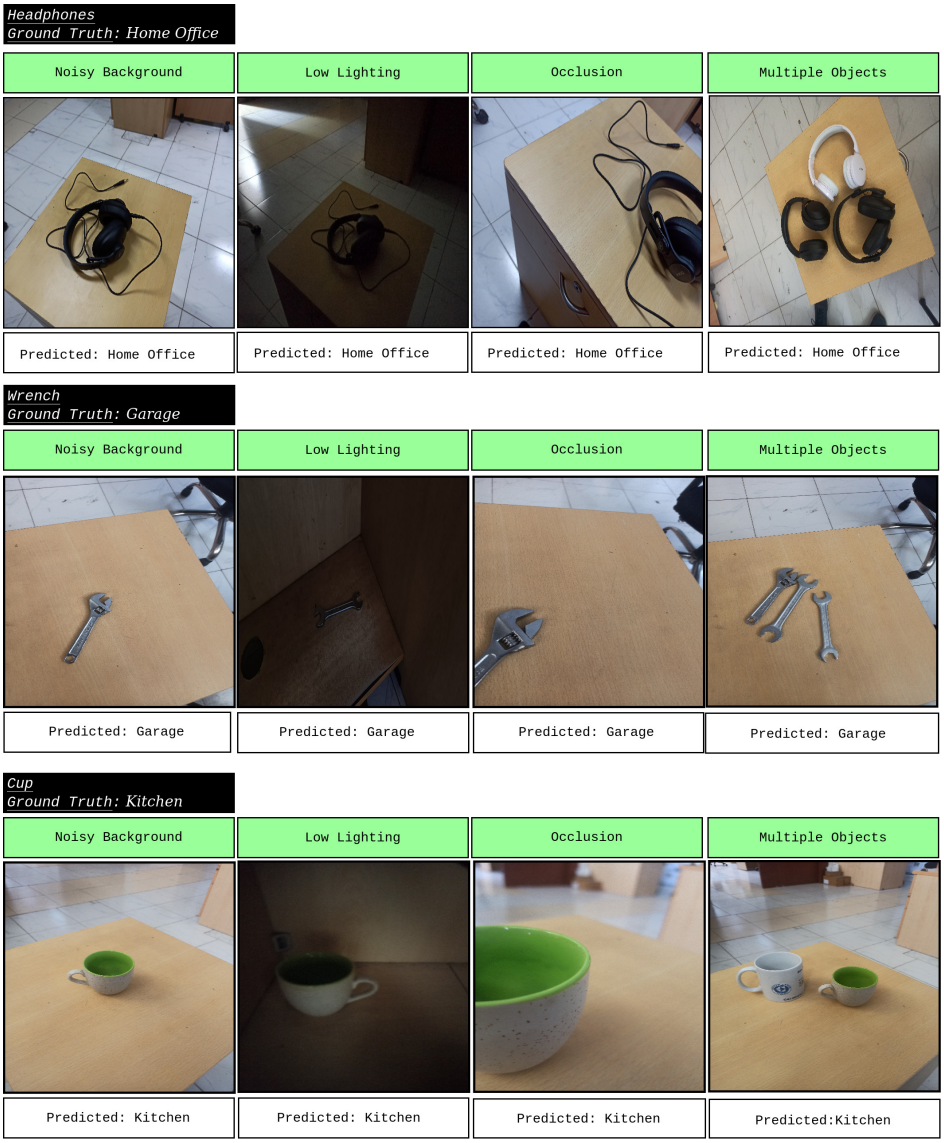


Figure 5: Qualitative results on real-world images show the usability of our model for detection of ideal location of objects based on visual inputs

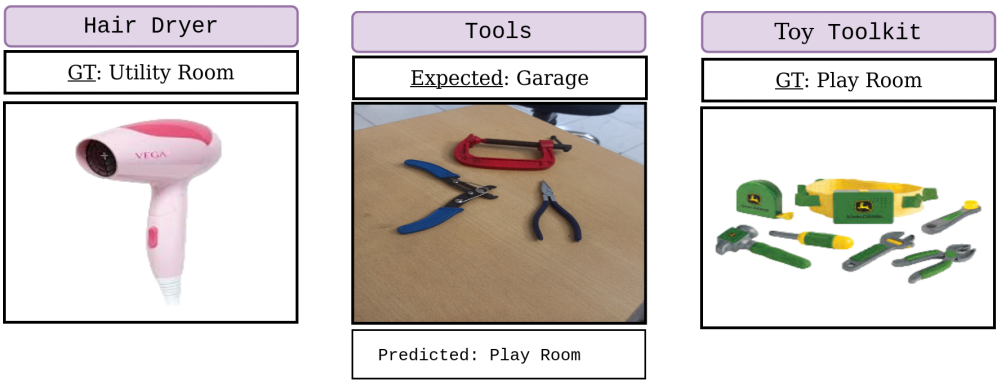
7.1 Success Cases

Figure 6: Success of the model on unseen images (absent in the training set)

7.2 Failure Cases

Figure 7 shows some specific limitations of our models. In Figure 7a 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 7b shows the limitation of the model in identifying real objects with similar category of toys.

(a) When testing on unseen object category, model confuses with similar looking images



(b) Failure in differentiating real-world objects with similar looking toys

Figure 7: Failure cases of our model

CLIP Model	Test mAP ⇑		Hit-Ratio ⇑					
	Before	After	Top-1		Top-3		Top-5	
			Before	After	Before	After	Before	After
ConvNext	0.405	**0.64**	0.223	**0.53**	0.472	**0.69**	0.632	**0.76**
ViT	0.456	**0.77**	0.256	**0.68**	0.576	**0.77**	0.710	**0.83**
RN50	0.453	**0.59**	0.275	**0.46**	0.546	**0.63**	0.643	**0.74**

Table 4: Pretrained CLIP language Features when passed through our GCN Encoder (trained to generate embeddings using CLIP Image Features) perform better than just CLIP Language Features without passing them through the GCN **H?**.

Lang Model	Test mAP ⇑		Hit-Ratio ⇑					
	Before	After	Top-1		Top-3		Top-5	
			Before	After	Before	After	Before	After
ConvNext	0.41	**0.73**	0.24	**0.62**	0.46	**0.81**	0.62	**0.88**
ViT	0.42	**0.85**	0.25	**0.76**	0.49	**0.93**	0.65	**0.97**
RN50	0.39	**0.66**	0.19	**0.53**	0.45	**0.75**	0.67	**0.81**

Table 5: Performance of CLIP Image Encoder Features without passing through GCN vs performance when passed through the trained GCN encoder **H?**.

8 Some additional Plots

8.1 t-SNE

Using T-distributed Stochastic Neighbor Embedding (t-SNE) to visualize the high-dimensional embeddings, we observe initial random nodes in 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 together as shown in Figure 9

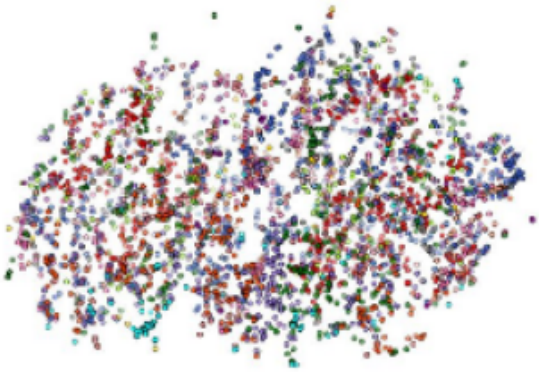


Figure 8: Untrained TSNE

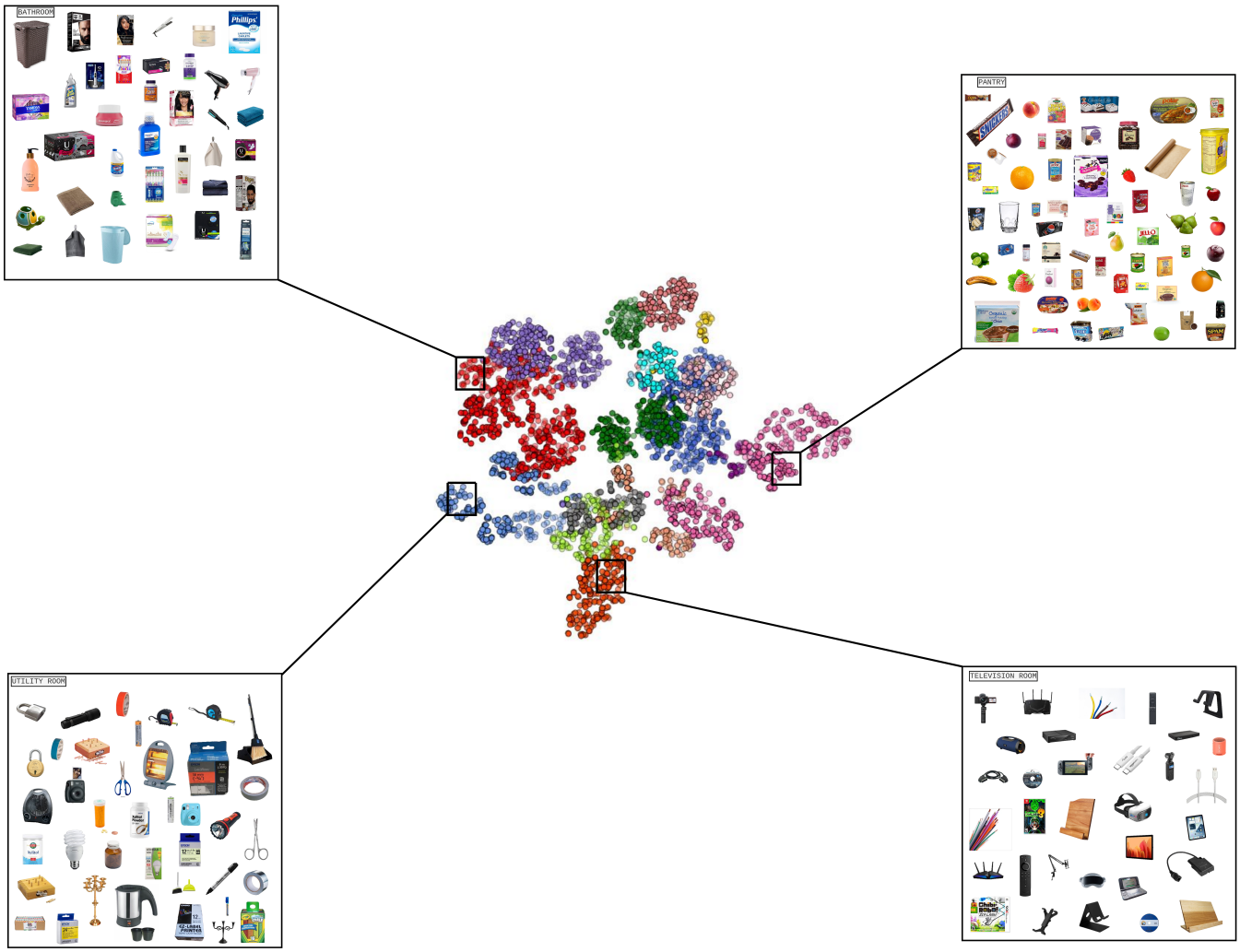


Figure 9: t-SNE visualization of our embeddings on the test split of the Web Scraped Dataset. The boxes show images of objects belonging to the same rooms getting clustered [a]

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