Is Large Language Model All You Need for 3D Scene Understanding

Weihan Xu, Ruoyu Wang, Xihang Yu

April 19, 2023

Abstract

Robotic applications rely on scene understanding to analyze objects within a 3D environment. One crucial component of scene understanding is semantics labeling, which involves assigning class labels to semantic regions based on the objects within them. A visualization of the semantics labeling process is shown in Figure 1. In a recent study [2], Large Language Models (LLMs) were found to be effective in incorporating common sense knowledge during the labeling process. In this project, we aim to compare two LLMs, GPT-J and RoBERTa, using fine-tuned feed-forward and contrastive networks, which were not evaluated in [2], for the semantic labeling task. The contributions of this project are twofold: (i) The proposed GPT-J with fine-tuned feed-forward network achieves state-of-the-art(SOTA) performance, and (ii) by varying the number of candidate objects, adopting ChatGPT-based room detection and fine-tuning a whole BERT-based network, we explore the possible performance bottleneck of our proposed GPT-J pretrained network. Open-source software is available for download at https://github.com/XihangYU630/llm_scene_understanding_gptj.

1 Problem Description

The understanding of semantic scenes is becoming increasingly important for robotics, particularly with regards to places classification within 3D scenes. Although existing algorithms such as Kimera[7] and Hydra [6] have produced positive results in 3D scene reconstruction, the nodes generated by these algorithms are unlabeled, i.e. whether a particular space is a bedroom or a kitchen remains unknown. These labels are essential for subsequent reasoning and planning tasks, as the robot needs to understand, for example, the meaning of an instruction that "Go to the kitchen and cook the egg" and match "kitchen" in the sentence with the place in the reconstructed map produced by Simultaneous localization and mapping algorithm (SLAM).

Previous work [2] in scene labeling, as far as we know, first utilized natural language tools for classifying rooms. In this paper, we use large language models like BERT and RoBERTa to produce high-dimensional sentence embeddings and compare the results of GPT-J with the methods proposed in the original paper. Our study achieves state-of-the-art performance of 67.20% accuracy on the test dataset by using GPT-J and a feed-forward network. However, since a real-world robot perception system should expect higher accuracy, we aim to explore possible performance bottlenecks in our proposed GPT-J pretrained network. To this end, we vary the number of candidate objects, adopt ChatGPT-based room detection, and fine-tune a whole BERT-based network. The results indicate that searching for bottleneck of performance is still an ongoing research and using images is a natural next step. In a short word, our research project contributes by demonstrating the effectiveness of GPT-J and exploring possible performance limitations in this context.

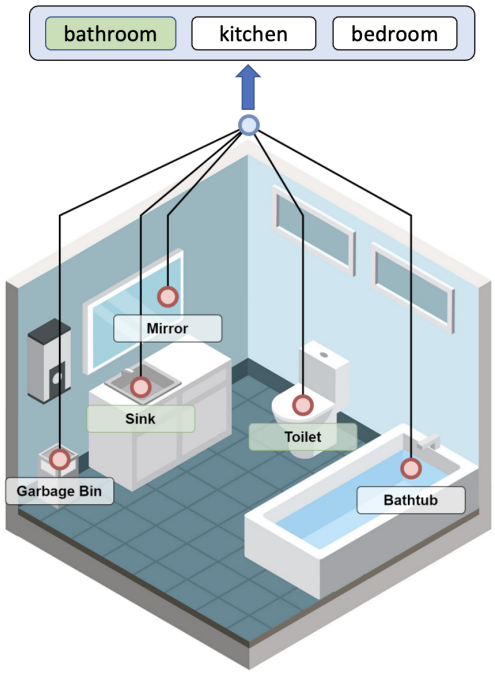

Figure 1: Illustration of place labeling task. Given low-level information, predict labels (rooms or buildings).

2 Related Work

2.1 3D Dynamic Scene Graphs

3D Dynamic Scene Graph (DSG) is a layered graph where nodes represent spatial concepts at different levels of abstraction, and edges represent spatio-temporal relations among nodes. [7] introduces Kimera, which is the first fully automatic method to build a DSG from visual-inertial data. Based on Kimera, [6] introduces Hydra, which is the first real-time spatial perception system that optimizes 3D scene graph in response to loop closure. It is used for 3D Scene Graph Construction and Optimization.

2.2 Semantic Labeling

[4] uses Bayesian probabilistic framework for determinig room labels and classifies based off the most common objects in a room. [2], utilizes natural language tools and introduces three paradigms for leveraging language for classifying rooms in indoor environments based on their contained objects, including zero-shot approach, a feed-forward classifier approach and a contrastive classifier approach.

2.3 Representation Learning on Graphs

[5] introduces GraphSAGE, which is a general inductive framework that leverages node feature information to efficiently generate node embeddings for previously unseen data. [9] introuduces Graph Attention Networks(GATs) is a novel neural network architectures that operates on graph-structured data, leveraging masked self-attentional layers to address the shortcoming of prior methods based on graph convolutions or their approximations.

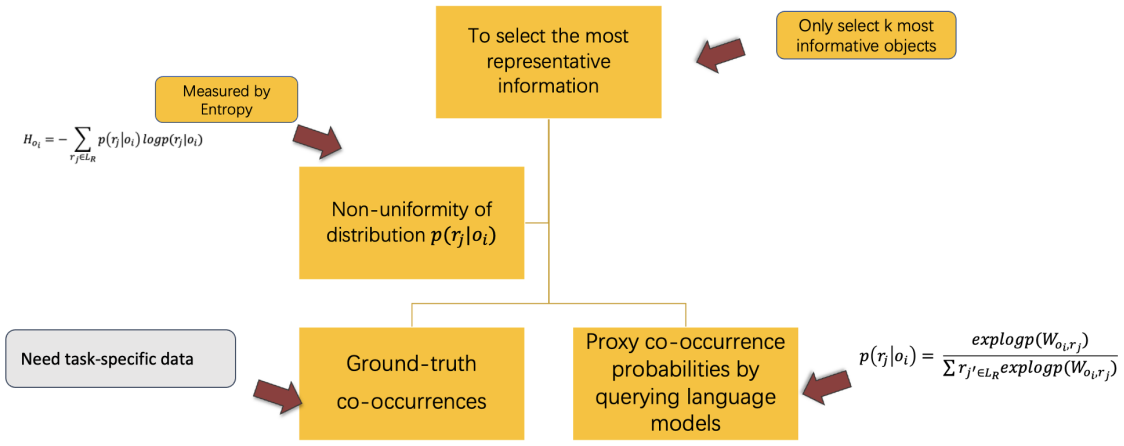

Figure 2: Illustration of data preprocessing.

3 Methodology

In this project, we are interested in determining the type of a room based on the objects in the room. In the remaining part of this section, we will first discuss dataset selection in Sec.3.1 and data preprocessing in Sec.3.2. Then feed-forward head network with GPT-J in Sec.3.3 and contrastive learning with GPT-J in Sec.3.4 will be discussed.

3.1 Dataset

We will evaluate the algorithms on scene graphs from Matterport3D dataset [1], which is used in [2] and other robot indoor navigation tasks. This dataset contains semantic labeled rooms (hence suitable for room classification task) and objects with mask segmentations and bounding boxes. We consider mpcat40 (35 object labels) and nyuClass (201 object labels) [8]. In total, there are 81 buildings with 1878 and 1866 rooms (for mpcat40 and nyuClass respectively), each with one of 23 room labels. Note that we only use mpcat40 labels for building classification task since mpcat40 is less informative and specific in terms of object labels as pointed out in [2]. It is important to take into consideration that the labeling of rooms in nyuClass and buildings in mpcat40 is biased due to the nature of Matterport3D. For example, labels such as bedrooms and houses are more common while labels such as lounges and office complexes are less common. The frequency distribution of the room labels is approximately shown in Figure 10, while the frequency distribution of the building labels is approximately shown in the column Total in Dataset of Table 4.

3.2 Data Preprocessing

As pointed out in [2], some of the objects might be shared across different type of rooms like chair and light, which might bring noise into the model. In order to get the most representative information, we selected only k most semantically informative objects as in [2]. A semantically informative object is defined as an object that only appear in only a few room types. This is measured by non-uniformity of probability $p(r_i|o_i)$, where $o_i \in L_o$ is the object label, $r_j \in L_R$ is the room label. We measure non-uniformity by entropy:

$$H_{o_i} = -\sum_{r_j \in L_R} p(r_j|o_i) \log p(r_j|o_i) \tag{1}$$

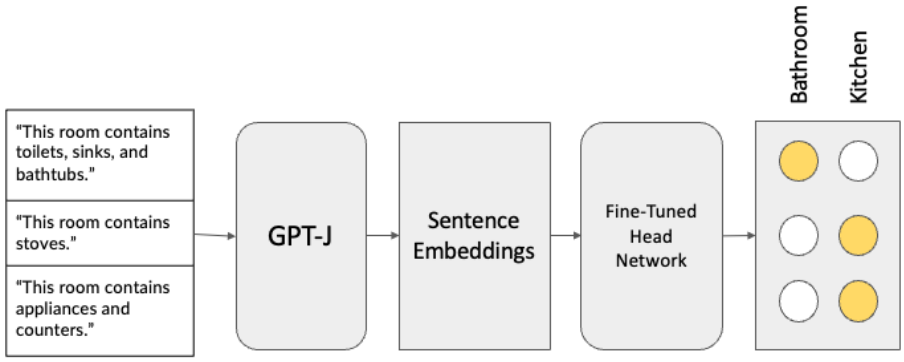

Figure 3: Examples of feed-forward network with GPT-J

If the distribution $p(r_i|o_i)$ is uniform the entropy is maximized and minimized when it is one-hot. Thus, we select top-k objects:

$$O_{best} = \underset{o_i}{argmin_k}[H_{o_i}] \tag{2}$$

The remaining question is how to proximate distribution of $p(r_j|o_i)$. Two approaches exist for computing $p(r_j|o_i)$. The first approach involves utilizing ground-truth co-occurrences. It is similar to how we extract Pointwise Mutual Information from a term-document matrix, which displays the co-occurrence of a term and a document. In this method, we calculate how frequently each object label appears in each room type and normalize it across rooms. To avoid requiring task-specific data, we employ Laplace smoothing.

The second approach entails utilizing language models to estimate the probability of co-occurrence between a room label and an object label. To achieve this, we can use a language model to anticipate the likelihood of a sentence containing both the object label and the room label. Specifically, we use

$$p(r_j|o_i) \approx \frac{\exp \log p(W_{o_i,r_j})}{\sum_{r_{j'} \in L_R} \exp \log p(r_{j'}|o_i)} \tag{3}$$

where W_{o_j,r_j} is the query string "A room containing o_i is called a(n) r_j." and $\log p(W_{o_i,r_j})$ is computed via language model.

As shown in Figure 2, we take the n most informative objects in each room and find all k-object permutations, producing P_k^n query data points per room of the form "This room contains o_1, o_2, ... and o_k.", all of which correspond to the room's label.

We do this for $(k, n) \in \{(1, 1), (1, 2), (2, 2), (2, 3), (3, 3), (3, 4)\}$.

3.3 Feed-forward Network with GPT-J

We investigate the use of large language models to impart common sense for scene understanding. As shown in Figure 3, we fine-tune a feed-forward network. In particular, the objective is to find the label giving us the maximum value of the output.

$$\hat{r} = \underset{r_j}{argmax}[f_\theta(\Lambda(W))]_{r_j} \tag{4}$$

W is a query of the form "This room contains o_1, o_2, ... and o_k." $\Lambda(W)$ takes in a sentence from query space and outputs to a hidden space. A neural network $f_\theta : \Lambda(W) \rightarrow R^{|L_R|}$ takes in query embeddings and outputs prediction logits.

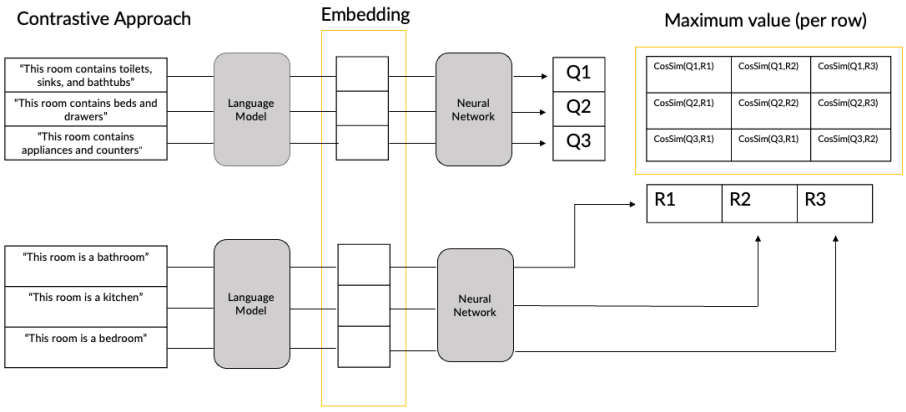

Figure 4: Examples of contrastive network with GPT-J

3.4 Contrastive Network with GPT-J

As shown in Figure 4, we fine-tune a contrastive network using the hidden output of GPT-J. The objective is to find the room label whose embedding has the highest cosine similarity with that of the query:

$$\hat{r} = \underset{r_j}{argmax}[g_{\theta_1}(\Lambda(W)) \cdot h_{\theta_2}(\Lambda(W^R_{r_j}))] \tag{5}$$

where W is a query of and string $W^R_{r_j}$ is a natural language template in the form of "This room is a(n) r_j." $\forall r_j \in L_R$. Note that $g_{\theta 1}, h_{\theta 2}$:$\Lambda(W) \rightarrow \{\hat{v}|\hat{v} \in R^N, \|\hat{v}\|_2 = 1\}$ are two shallow multi-layer perceptrons neural networks that mapping language embeddings into the same space.

4 Experiments

4.1 Baseline

To establish a basis for comparison, we selected two fundamental algorithms. The first algorithm we chose was the RoBERTa pre-trained model equipped with a feed-forward/contrastive head network because it exhibited the best performance among all the methods discussed in [2]. Additionally, we decided to utilize the zero-shot GPT-J to demonstrate that with a feed-forward/contrastive head network, GPT-J can effectively learn from data. It's worth mentioning that the second algorithm was previously tested in [2].

4.2 Experimental Setup

4.2.1 Pre-trained GPT-J with Head Network

Table illustrates the parameters employed in our experiments with the feed-forward head network (left table) and the contrastive head network (right table). We use 50/20/30 train/validation/test split in nyuClass for all methods except for zero-shot GPT-J, where only test split is used to make it comparable with other methods. Table 2 shows the implementation details of the training process. It is worth noting that we did not utilize a training epoch as high as 200 for the baseline, as in the original paper, since the training loss stabilized after just 10 epochs (or 20 epochs for RoBERTa+contrastive). As for the zero-shot GPT-J baseline, we generated query sentences using three most informative objects. Additionally, we tested both the ground truth and proxy co-occurrence probability for all methods, including both our proposed methods and the baselines. For building trials, we divide the buildings in mpcat40 dataset into a 40/20/40 train/validation/test split. Table 3 shows the implementation details of the training process for building detection. We

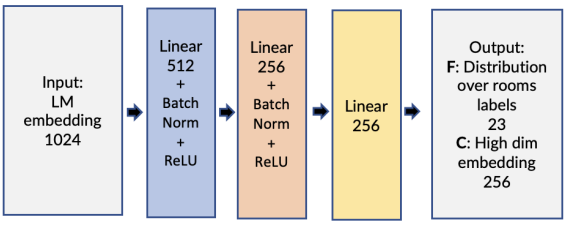

Figure 5: Head network architecture for RoBERTa pretrained model: The fine-tuned feed-forward and contrastive heads use the same architecture, but with different output dimensions (reported under F: and C: respectively).

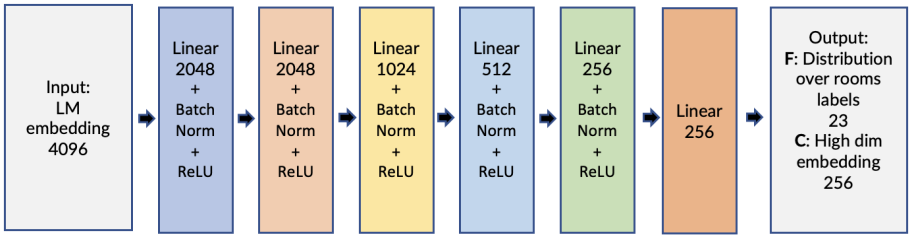

Figure 6: Head network architecture for GPT-J pretrained model: The fine-tuned feed-forward and contrastive heads use the same architecture, but with different output dimensions (reported under F: and C: respectively).

use the same set of parameters as reported in [2] for GPT-J + feedforward network except for reduced number of epochs. The different high-level sentence embeddings of the RoBERTa and GPT-J pretrained models necessitated the use of different network architectures. Specifically, the network model with the RoBERTa pretrained model used the architecture shown in Figure 5, while the GPT-J network model used the architecture shown in Figure 6. It should be noted that the contrastive head network for RoBERTa had a slightly different architecture from the original architecture in [2] in terms of the number of neurons in certain layers, as we found our model performed better with this adjustment. However, we kept the same architecture with [2] for the feed-forward network for RoBERTa.

4.2.2 ChatGPT

In November 2022, OpenAI launched ChatGPT, an AI chatbot that uses the GPT-3.5 and GPT-4 families of large language models (LLMs) and has been fine-tuned using a combination of supervised and reinforcement learning methods. ChatGPT has demonstrated high performance in various tasks. For this study, we selected 217 data points from the test dataset, each containing no more than three objects for prediction. We used the strategy outlined in Sec.3.2 to select objects based on their GT co-occurrence probability. After selecting the objects, we constructed a query sentence in the form of "W = 'This room contains (objects). What is the most likely label for this room among bar, bathroom, bedroom, classroom, closet, conference room, dining room, family room, game room, garage, gym, hallway, kitchen, laundry room, library, living room, lobby, lounge, office, spa, staircase, television room, utility room?'". We then tasked ChatGPT with predicting the most likely room label given the information about the objects.

4.2.3 BERT fine-tuning

For the BERT fine-tuning experiment on the nyuClass dataset, a train/validation/test split of 40/20/40 is used. To generate queries for each room, the four most significant objects are selected and $P_3^4 = 24$ different queries are created by generating all possible permutations of these objects.

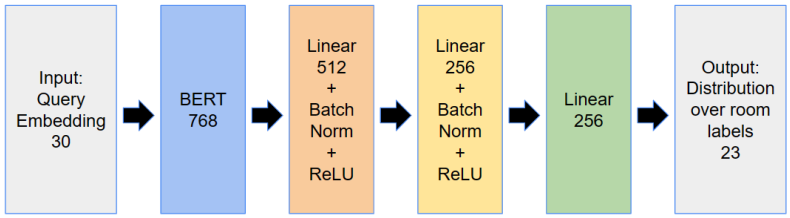

Figure 7: Network architecture for BERT classifier.

The queries are tokenized using BertTokenizer and encoded as a list of integers for input into the BERT model. To ensure consistency in length, all queries are padded to a length of 30 using max-padding. The embedded queries are then fed into the BERT classifier model (as shown in Figure 7), which is fine-tuned for 100 epochs using cross-entropy loss and Adam optimizer with a learning rate of 0.0001 and weight decay of 0.001. The batch size used is 128, and the fine-tuned model is tested on the test dataset and compared to the baseline.

4.3 Evaluation

We will be using classification accuracy as the evaluation metric for the experiments involving pre-trained GPT-J/RoBERTa with fine-tuning head network, zero-shot GPT-J, and fine-tuning BERT. The accuracy will be calculated using the following formula:

$$Accuracy = \frac{\#\text{Successful Room Predictions}}{\#\text{All Room Predictions}} \tag{6}$$

For the zero-shot ChatGPT experiment, we will be computing macro/micro averaged precision/recall/F1. Macro-averaging compute performance for each class and then average. Micro-averaging collects decisions for all classes and compute contingency table to get the result. The equations for calculating precision and recall are as follows:

$$Precision = \frac{\#\text{True Positive}}{\#\text{True Positive} + \#\text{False Positive}} \tag{7}$$

$$Recall = \frac{\#\text{True Positive}}{\#\text{True Positive} + \#\text{False Negative}} \tag{8}$$

$$F1 - score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{9}$$

5 Results

By replacing RoBERTa-large with GPT-J and our approach achieves SOTA performance. Quantitative results will be presented in Sec.5.1. As described in Sec.5.2, to test the generalizability of the proposed method, we also test it for building type detection task and achieve better result than baseline. However, with testing accuracy that is lower than 70% in room detection task, it is not possible to deploy the room detection algorithm on real world robotics systems. Therefore, potential bottleneck is explored in the following sections. To explore possible bottleneck of performance, varying number of most informative candidate objects in Sec.5.3, ChatGPT-based zeroshot room detection in Sec.5.4 and finetuning BERT network in Sec.5.5 are tested and quantitative results are presented.

Table 1: Accuracies in room prediction for all approaches. As haven been discussed in Sec.3.2, GT represents using ground truth to select the informative objects while Proxy queries language model to proximate co-occurrency probability.

Zero-shot		Feed-forward				Contrastive			
GPT-J(Baseline)		RoBERTa(Baseline)		GPT-J(Ours)		RoBERTa(Baseline)		GPT-J(Ours)	
GT	Proxy	GT	Proxy	GT	Proxy	GT	Proxy	GT	Proxy
45.52%	25.27%	66.20%	56.94%	**67.20 %**	57.14%	64.19%	54.73%	62.58%	53.92%

Table 2: Hyperparameters used in room prediction. Left table shows the parameters used in feed-forward network. Right table shows the parameters used in contrastive network.

Hyperparameters	RoBERTa(Baseline)	GPT-J(Ours)
Loss Function	Cross-entropy	Cross-entropy
Optimizer	Adam	Adam
Learning Rate	0.0001	0.0001
Weight Decay	0.001	0.001
Step Size	10	10
Gamma	0.5	0.5
Epochs	10	10
Batch Size	512	512

Hyperparameters	RoBERTa(Baseline)	GPT-J(Ours)
Loss Function	Cross-entropy	Cross-entropy
Optimizer	Adam	Adam
Learning Rate	0.00001	0.00001
Weight Decay	0.001	0.001
Step Size	20	20
Gamma	0.9	0.9
Epochs	20	10
Batch Size	512	512

5.1 Pre-trained GPT-J with Head Network

As we can see in Table 1, feed-forward/contrastive + RoBERTa/GPT-J are tested. The accuracies on testing dataset are recorded. As shown in the table, methods with GT co-occurency probability consistently outperform methods with proxy co-occurency probability. This is within our expectation since, since by using GT probability better, training process better fits into a specific problem. We achieve SOTA performance by using GPT-J + feed-foward with ground truth object query at 67.20%. Notice that the baseline here achieves best performance in [2]. In other words, the result of GPT-J + feed-forward is the best ever record.

5.2 Building Type Classification using GPT-J

With generalizability in mind, we will evaluate the performance of building prediction using fine-tuning feed-forward and contrastive model with GPT-J or RoBERTa. Given the rooms contained in a building, the network infers the building containing the rooms. Specifically, it classifies a query W "This building contains r1, r2, ... and rk." into three classes {house, office complex, spa resort}. only feed-forward + RoBERTa/GPT-J are tested and results are shown in Table 4. Again, GPT-J + feed-forward achieves slightly better results in building classification task.

Table 3: Hyperparameters used in building prediction. Table shows the parameters used in feed-forward network.

Hyperparameters	RoBERTa(Baseline)	GPT-J(Ours)
Loss Function	Cross-entropy	Cross-entropy
Optimizer	Adam	Adam
Learning Rate	0.0001	0.0001
Weight Decay	0.001	0.001
Step Size	20	20
Gamma	0.99	0.99
Epochs	100	10
Batch Size	512	512

Table 4: Accuracies in building prediction for RoBERTa/GPT-J+feed-forward. Note that only feed-forward networks are compared. As haven been discussed in Sec.3.2, GT represents using ground truth to select the informative objects while Proxy queries language model to proximate co-occurency probability.

Building labels	RoBERTa(Baseline)	GPT-J(Ours)	Total in Dataset
House	27	26	27
Office Complex	1	2	2
Spa Resort	0	1	4
Total	28(84.85%)	**29(87.88%)**	33

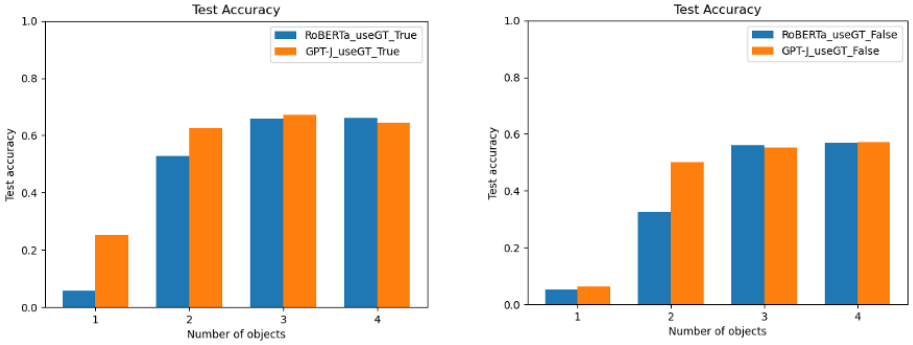

Figure 8: Test dataset accuracy results of varying the number of candidate objects in query sentence for feed-forward head network.

5.3 Varying Number of Candidate Objects

In this experiment, we are exploring the impact of the number of candidate objects on the accuracy of various methods. To do so, we varied the number of candidate objects, denoted by n, from 1 to 4. For each value of n, we generated query sentences by doing all permutations of the candidate objects as described in Sec.3.2. Specifically, for $n = 1$, we generated query sentences with one candidate object, denoted by P_1^1, for $n = 2$, we generated query sentences with two candidate objects, denoted by P_k^n where $(k, n) \in \{(1, 1), (1, 2), (2, 2)\}$, for $n = 3$, we generated query sentences with three candidate objects, denoted by P_k^n where $(k, n) \in \{(1, 1), (1, 2), (2, 2), (2, 3), (3, 3)\}$, and for $n = 4$, we generated query sentences with four candidate objects, denoted by P_k^n where $(k, n) \in \{(1, 1), (1, 2), (2, 2), (2, 3), (3, 3), (3, 4)\}$.

Tables 1 and 4 demonstrate that using GPT-J instead of RoBERTa does not result in significant improvements. The accuracies of nearly all the methods remain at around 60%. One hypothesis for this performance bottleneck is the number of candidate objects in the query sentence. To explore this hypothesis, we increased the number of candidate objects in the query sentence and tracked the corresponding accuracies on the test dataset. Figures 8 and 9 depict these results. Two key observations emerged: (1) Increasing the number of candidate objects can enhance accuracy, and (2) GPT-J consistently outperforms RoBERTa for both ground truth/proxy and feed-forward/contrastive methods. However, most of the performance benefits occur when the number of candidate objects is less than 3. Accuracy convergence is observed when the number of candidate objects is 4, with an accuracy of about 60% for all methods. This observation suggests that the number of candidate objects may not be the bottleneck as the number of candidate objects increases, even though it may impact the accuracy for small number of candidate objects.

5.4 Zero-shot ChatGPT

The results of the zero-shot evaluation show a micro precision of 0.4468 and a micro recall of 0.4264, resulting in an F1 measure of 0.4364. On the other hand, the macro precision is 0.390 and the macro recall is 0.3876, resulting in an F1 measure of 0.3502. It is important to note that the

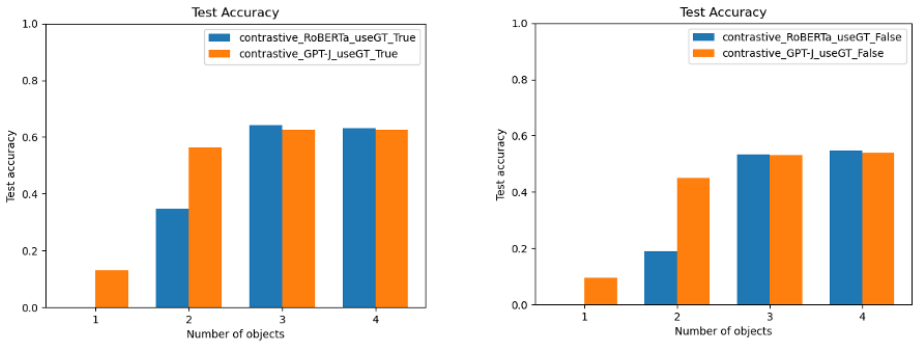

Figure 9: Test dataset accuracy results of varying the number of candidate objects in query sentence for contrastive head network.

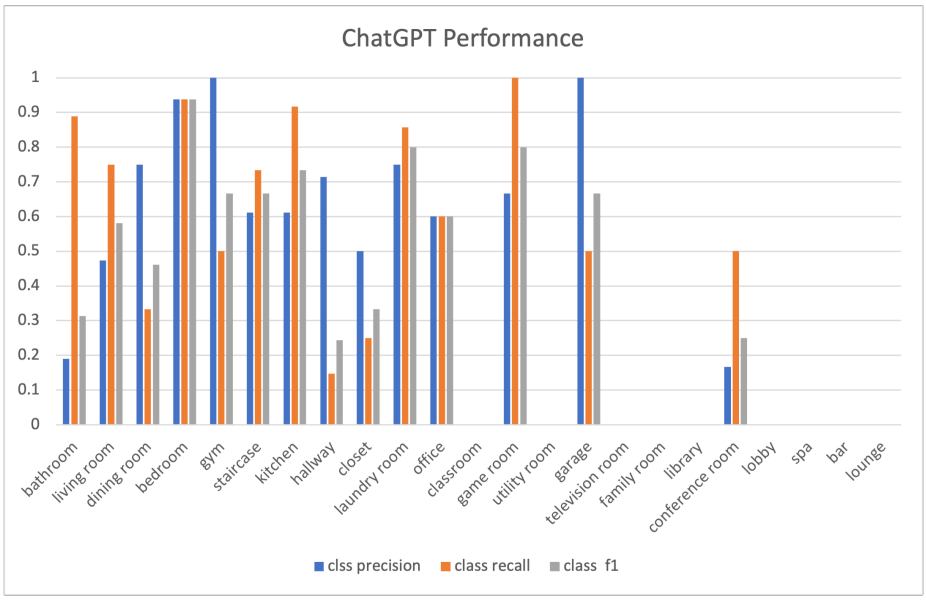

Figure 10: ChatGPT Performance

micro scale has a better result, suggesting that the class label distribution is imbalanced. To ensure comparability, we selected the same class labels as [2]. The zero-shot ChatGPT method performed better than the zero-shot GPT-J method, with an accuracy of 48.83% compared to 45.52%. However, the accuracy is still smaller than most baselines and proposed methods. The detailed results are presented in Figure 5.4. We observed three interesting patterns in the results:

1. High Precision but Low Recall: dining room, gym, hallway, closet, garage. The model predicts those room type conservatively. One of the reason we can think of is that those room types might lack disambiguating objects and ChatGPT need further information to do the predictions.

2. High Recall but Low Precision: bathroom, living room, staircase, kitchen, laundry room, coference room. One of the reason might be that in the training datset of ChatGPT, there were some object and room pairs linked more common than those in our test dataset. Therefore, ChatGPT did the prediction aggressively.

3. Hight Precision and High Recall: bedroom, office. Bedroom and office have objects almost exclusive to them. For example, office has monitors and desks. Bedroom has beds and pillows.

Table 5: Accuracies in room prediction for BERT fine-tuning. As haven been discussed in Sec.3.2, GT represents using ground truth to select the informative objects while Proxy queries language model to proximate co-occurrency probability.

Zero-shot		Feed-forward		Contrastive		BERT Fine-tuning (Ours)	
GPT-J(Baseline)		RoBERTa(Baseline)		RoBERTa(Baseline)			
GT	Proxy	GT	Proxy	GT	Proxy	GT	Proxy
52.20%	27.24%	66.20%	56.94%	64.19%	54.73%	59.35%	53.52%

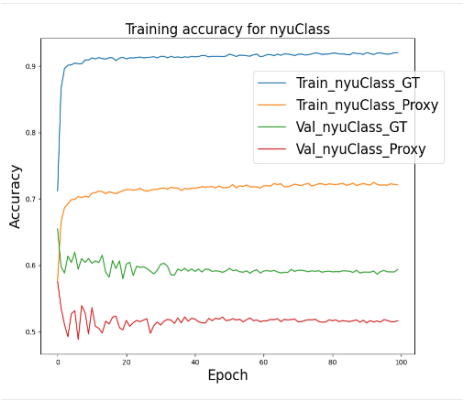

Figure 11: Training accuracy for fine-tuning BERT

5.5 BERT Fine-tuning

The performance test results of the fine-tuned BERT classifier are presented in Table 5. The results show that the fine-tuning of BERT did not improve the performance compared to the baseline except for the zero-shot GPT-J or the proposed GPT-J+feed-forward. This could be due to the small size of the dataset used in the training process, which could have caused overfitting on the training data. Figure 11 shows that the training accuracy reaches as high as 90% for GT co-occurrence probability after five epochs, but the validation accuracy initially declines and remains at a low level.

6 Conclusions

This project achieved state-of-the-art (SOTA) performance by replacing the RoBERTa-large pretrained model with GPT-J. Despite the marginal improvement, the performance bottleneck still exists and searching for it remains an ongoing research topic. One possible solution is to enlarge the training dataset and finetune a large language model, similar to the method discussed in Sec.5.5. This approach can leverage the scaling power of a large language model and prevent overfitting. Another potential future direction is to use GPT-based networks, which incorporate common sense knowledge during training, but most GPT models are not currently open-sourced. Multi-modal room classification is also a promising area for future research, where not only semantic labels but also non-semantic visual information, such as geometric structures and object relations, are considered, similar to how humans classify rooms. As multi-modal pre-trained models like PaLM-E [3] and GPT-4 become available, it is expected that a generalizable robot spatial perception system will be closer to reality than ever before.

References

[1] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in

indoor environments. *arXiv preprint arXiv:1709.06158*, 2017. 3

[2] William Chen, Siyi Hu, Rajat Talak, and Luca Carlone. Leveraging large language models for robot 3d scene understanding. *arXiv preprint arXiv:2209.05629*, 2022. 1, 2, 3, 5, 6, 8, 10

[3] Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. Palm-e: An embodied multimodal language model. In *arXiv preprint arXiv:2303.03378*, 2023. 11

[4] David Fernandez-Chaves, Jose-Raul Ruiz-Sarmiento, Nicolai Petkov, and Javier Gonzalez-Jimenez. From object detection to room categorization in robotics. In *Proceedings of the 3rd international conference on applications of intelligent systems*, pages 1–6, 2020. 2

[5] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017. 2

[6] Nathan Hughes, Yun Chang, and Luca Carlone. Hydra: a real-time spatial perception system for 3d scene graph construction and optimization. 2022. 1, 2

[7] Antoni Rosinol, Andrew Violette, Marcus Abate, Nathan Hughes, Yun Chang, Jingnan Shi, Arjun Gupta, and Luca Carlone. Kimera: From slam to spatial perception with 3d dynamic scene graphs. *The International Journal of Robotics Research*, 40(12-14):1510–1546, 2021. 1, 2

[8] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. *ECCV (5)*, 7576:746–760, 2012. 3

[9] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio, et al. Graph attention networks. *stat*, 1050(20):10–48550, 2017. 2