Applications of Large Language Models for Robot Navigation and Scene Understanding

by

William Chen

S.B., Electrical Engineering and Computer Science,
Massachusetts Institute of Technology (2022)

Submitted to the
Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2023

© 2023 William Chen. All rights reserved.

The author hereby grants to MIT a nonexclusive, worldwide, irrevocable, royalty-free license to exercise any and all rights under copyright, including to reproduce, preserve, distribute and publicly display copies of the thesis, or release the thesis under an open-access license.

Authored By: William Chen
Department of Electrical Engineering and Computer Science
May 12, 2023

Certified by: Luca Carlone
Associate Professor, Department of Aeronautics and Astronautics
Thesis Supervisor

Accepted by: Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Applications of Large Language Models for Robot Navigation and Scene Understanding

by

William Chen

Submitted to the Department of Electrical Engineering and Computer Science
on May 12, 2023, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Common-sense reasoning is a key challenge in robot navigation and 3D scene understanding. Humans tend to reason about their environments in abstract terms, with a wealth of common sense on object and spatial relations to back up such inferences. Thus, if robots are to see widespread deployment, they must also be able to reason with such knowledge to support tasks specified in such terms. As modern language models trained on large text corpora encode much worldly knowledge, we thus investigate methods for extracting common sense from such models for use in non-linguistic semantically grounded robotics tasks. We start by examining how language models can be used for attaching abstract room classes to locations based on visual percepts and lower-level object classes, commonly generated by spatial perception systems. We detail three language-only approaches (zero-shot, embedding-based, and structured language) as well as two vision-and-language approaches (zero-shot and fine-tuned), finding that language-leveraging systems outperform both standard pure-vision and scene graph neural classifiers while yielding impressive generalization and transfer abilities. We then consider a simple robot semantic navigation task to see how an agent can act upon prior knowledge encoded within language models in order to find goal objects by reasoning about where such objects can be found. Our framework, Language Models as Probabilistic Priors (LaMPP), uses the language model to fill in parameters of standard probabilistic graphical models. We also touch upon use cases outside of robotics, namely semantic segmentation and video action segmentation. Lastly, we show how common-sense knowledge can be extracted from language models and encoded in abstract spatial ontology graphs. We measure how well language model scores align with human common sense judgements regarding object and spatial relationships. Ultimately, we hope this work paves the way for more advanced robot semantic scene understanding and navigation algorithms that leverage language models.

Thesis Supervisor: Luca Carlone
Title: Associate Professor, Department of Aeronautics and Astronautics

Acknowledgments

First and foremost: I would like to thank my advisor, Luca Carlone – for guiding me through this thesis; for teaching me research skills, both technical and communicative; for giving me the chance to help teach the best class at MIT; for helping me uncover my passion for robotics and natural language processing; and, most of all, for answering my Slack messages at weird hours of the night. I'm sure I could continue this list for a lot longer, but all those words still would not adequately communicate my gratitude.

I'd also like to thank Jacob Andreas for both supervising the LaMPP project and being an invaluable source of mentorship and knowledge of natural language processing and its connections to robotics, autonomy, and decision-making. Your classes, advice, and discussions have really helped me shape my research interests, both throughout this project and going forward.

Next, I want to thank my paper collaborators and co-authors: Siyi Hu, Rajat Talak, Belinda Shi, Pratyusha Sharma, Jared Strader, and Nathan Hughes. These projects would not have been possible without all of your boundless talent and understanding. A big thank-you to the rest of SPARK Lab and the LINGO group as well.

Outside of my research colleagues, I want to thank all the friends, classmates, and professors I've met at MIT. Thank you all for constantly inspiring me to learn new things, face new challenges, and improve myself. The road ahead is still long, but I'll be sure to hold onto the lessons you all have taught me.

Finally, I'd like to thank my plush wolf, Steven Derpwolf, for his constant love and support. Gonna need a lot more of that for the adventures that await.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 The Path Towards Everyday Robots

Recent advances in artificial intelligence and machine learning have allowed robots to see a surge in deployment in both the domestic and industrial domains. Robots are now being used to monitor factories, fulfill shipping orders, and vacuum homes, among a myriad of other applications. However, there are still many obstacles to overcome. If robots are to see widespread adoption, then they must be able to not only map and localize in many environments, but also have a semantic understanding of said environments and the entities within them. Subsequently, they should also have the ability to act upon this understanding. For example, if a robot is told to "fetch a spoon," it should know that spoons are typically found within kitchens, which contain things like refrigerators and stoves. Then, based on this knowledge, it should be able to find its way to locations that are likely to be kitchens (for instance, based on an internal map representation).

These aspects are typically inferred using metric-semantic simultaneous localization and mapping (SLAM) algorithms, wherein a robotic agent maps its environment, determines its location within it, and annotates the map with semantic and spatial information [12]. Modern spatial perception systems, like Kimera [76] and Hydra [36], arrange this data in 3D scene graphs – data structures wherein nodes represent locations and entities (e.g., buildings, rooms, and objects), while edges represent spatial relationships.

Nodes can hold geometric information (like position and bounding box) for entities

and places in the scene. However, attaching semantic labels to these nodes still remains a major open obstacle, especially for nodes corresponding to high-level spatial concepts, like rooms and buildings. To label a room node, the system must consider what objects are in the room (e.g., if it contains a stove, the room is likely a kitchen). This necessitates a "common-sense" mechanism to provide such knowledge.

1.2 Language Models and Robotics

One largely unexplored candidate method for imparting this common sense is by using language models. As they are trained on large text corpora, they capture some of the semantic information within said datasets – e.g., a language model may learn that the sentence "Bathrooms contain ___." is better finished with "toilets" rather than "stoves," thereby containing some of the common sense needed for scene understanding. Furthermore, being what [94, 19] famously called an "infinite use of finite means," language allows arbitrary common-sense queries to be compactly made and evaluated, including ones with novel concepts and entities. This is important for spatial perception, as *deployed robots will naturally come across many objects that engineers did not expect during development*. Being able to do inference over novel object types would thus be highly beneficial.

We thus explore a variety of ways for doing object- and visual-percept-driven room and building classification using pre-trained large language models. We consider each approach's strengths, failure-cases, and data requisites in order to see what language model-leveraging approaches are most appropriate to robot perception.

Once a semantically rich representation of the environment is made available to a robot, it now must have a way to act on this knowledge, integrating its semantic understanding into its decision-making process. We showcase a simple method for extracting common-sense probabilistic priors on object-room co-occurrences from language models *and* how such information can be made actionable in embodied household object-finding tasks.

Lastly, we demonstrate the scalability of such language model methods in scene understanding by constructing an abstract representation of plausible spatial relations between different objects and locations. We discuss how this data structure can be automatically

generated and its possible uses in supplementing downstream learning tasks.

We now motivate each of the above directions in more detail.

1.3 Scene Understanding

While significant advances have been made in spatial perception, robots are still far from having the common-sense knowledge about household objects and locations of an average human. We would like them to have such understanding, as many tasks one might want a robot to do are either best specified in terms of an abstract location label ("Go clean the bedroom.") or would be aided by knowledge of such labels and what said locations typically contain.

We thus investigate the use of large language models to impart common sense for scene understanding in Chapter 2. Specifically, we introduce three paradigms for leveraging language for classifying rooms in indoor environments based on their contained objects: (i) a zero-shot approach, (ii) a feed-forward classifier approach, and (iii) a contrastive classifier approach. These methods operate on 3D scene graphs produced by modern spatial perception systems. We then analyze each approach, demonstrating notable zero-shot generalization and transfer capabilities stemming from their use of language. Finally, we show these approaches also apply to inferring building labels from contained rooms and demonstrate our zero-shot approach on a real environment. All code can be found at `https://github.com/MIT-SPARK/llm_scene_understanding`.

1.4 Robot Navigation

As with scene understanding, robot navigation is likewise made more efficient with the aid of common-sense prior knowledge, especially in structured household environments. Rather than using this common sense for constructing internal representations of the environment that are conducive to relevant inference tasks, in Chapter 3, we investigate how common sense from language models can be used for *decision-making*. That is, we look at how this knowledge can be integrated with lower-level robot control policies to improve

their performance on semantic navigation tasks.

The specific task we consider is the ObjectNav challenge, held in the Habitat AI simulator – a yearly competition wherein teams build and train agents to find goal objects in virtual home environments (constructed from 3D scans of actual homes) [98, 59, 90]. We alter the task to provide the robot with the high-level labels of rooms. With this abstract map information, we show how the agent can leverage language model judgements regarding which objects are often found in each location to more accurately and efficiently locate specified target objects.

This work was done as part of a wider project to investigate methods for extracting and using probabilistic priors on common-sense facts and relations distilled within large language models. This broad paradigm, which we call **La**nguage **M**odel **P**robabilistic **P**riors (or **LaMPP**), allows for the principled insertion of prior knowledge from language models into domain-specific generative probabilistic models. We show how such methods can similarly be applied to image semantic segmentation and video-action segmentation – all grounded tasks that language models do not receive direct exposure to. While we briefly cover these topics in Chapter 3, we primarily focus on LaMPP's applications to embodied/robotic navigation.

1.5 Large-scale Spatial Ontology

The first two works attempt to extract and distill scene common sense present within language models on the problem of room and building classification while the latter showcases how these models can provide actionable priors that improve robot navigation tasks. However, these are both still relatively small-scale instantiations of language models' possible uses in providing worldly common sense. Humans can reason about a much larger set of objects and locations, including outdoor ones that are of much grander scale.

We therefore investigate methods of creating large data structures that represent common-sense object-location relationships with the help of language models. In particular, we develop a **spatial ontology** – a graph structure wherein nodes represent abstract object and location concepts and edges represent common spatial relations between these concepts.

All such components are ungrounded in any particular scene; they describe how locations and things are *usually* related, as understood by human common sense.

We investigate how such information can be extracted from language models in a manner akin to how LaMPP's probabilistic priors are extracted. We then consider how this data can be parsed into our desired data structure, developing numerical tests to both measure and ensure that the ontology's representations are *aligned with human judgements*. We also consider the use of some recent advances in instruction-following language models, tuned with reinforcement learning with human feedback (RLHF) for the same ontology-construction task. Finally, we discuss some future works in this area: namely, (i) how to actually leverage this ontology for learning to perform abstract scene understanding on grounded scenes and (ii) how to further automate the ontology construction process.

1.6 Thesis Structure

In this thesis, we present each of the aforementioned works. Specifically, in Chapter 2, we discuss the use of language for abstract scene understanding. In Chapter 3, we look at its use case in robot navigation. Lastly, in Chapter 4, we consider how common sense in language models can be used to construct spatial ontologies.

As they are all heavily intertwined, we start each chapter off by discussing how the work ties into both the ones before it and the overall theme of using language models to guide robotics tasks that require abstract common sense. Then, we discuss related works in each areas. Last, we discuss the methods, experimental specifics, and results, as well as any implications or findings in the three works. Said projects have a lot of additional information or details, which we document in the appendices (which are referenced throughout).

After the projects, we conclude with a few remarks on future works, providing some thoughts on upcoming directions for language and robotics research in Chapter 5.

Chapter 2

Scene Understanding

In this work, we consider ways to leverage pre-trained language models for abstract robot scene understanding tasks – specifically, room classification. We show how such models provide useful priors that allow for efficient zero-shot or fine-tuned performance. We find that a structured-language approach wherein a language model is fine-tuned for room classification based on strings detailing spatial and low-level semantic object information outperforms similar language-based approaches that only consider object labels (and discards quantitative spatial information, due to it being difficult to express in natural language).

These experiments only reason about language descriptions, discarding visual information. However, such features still provide a useful semantic signal, so we also consider methods for using modern visual language models for robot scene understanding. We find that visual-linguistic approaches achieve better performance when combining vision with object labels in comparison to vision-only methods (while maintaining minimal hand-engineering effort). We therefore argue that combining modern semantic spatial perception systems with the common-sense priors found in pre-trained (visual) language models enables effective abstract scene understanding.

2.1 Related Works

Metric-Semantic SLAM: As high-level semantic understanding is vital for human-robot interaction and planning, there has been significant interest in combining classical methods

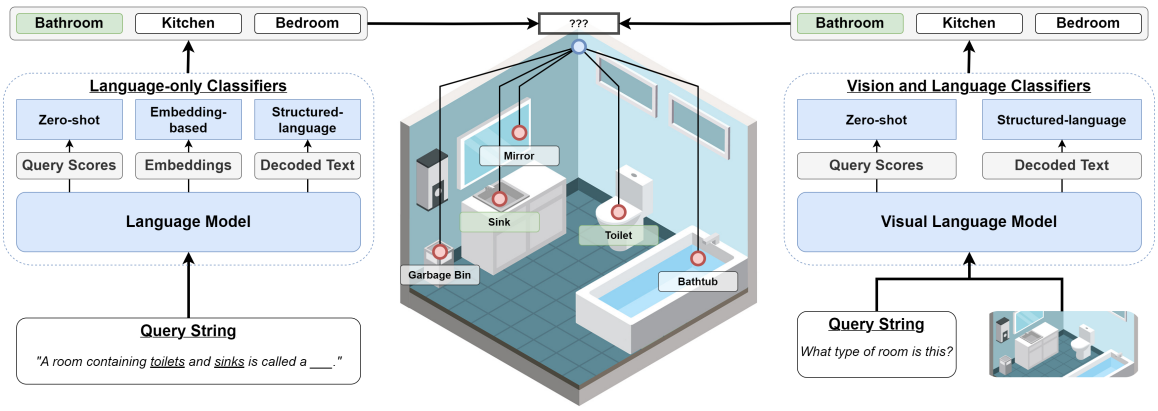

Figure 2-1: 3D scene graph example. We use (V)LMs to attach high-level labels to nodes using lower-level information.

with deep learning for metric-semantic SLAM. Such works generally focus on low-level representations, such as object-centric approaches [12, 23, 66, 67], dense approaches [7, 29, 77], or a hybrid of the two [50, 61]. However, these methods generally disregard higher-level semantic labeling which are needed for many planning and reasoning tasks.

Hierarchical Mapping: An alternative approach is to consider hierarchical maps, which represent the robot's environment at different levels of abstraction. Works like [78, 26] divide robot knowledge into spatial and semantic information, then anchoring the former to the latter. Our work focuses on scene graphs as a hierarchical map representation. Such data structures were first commonly used for visual relation tasks [56, 39], but have since been generalized to 3D, where they have found success in robotics [5, 75, 76, 36]. Nevertheless, anchoring semantics to spatial information in 3D scene graphs remains difficult, motivating us to look towards language models.

Language and Robotics: Using language models for robotics has been a rapidly-expanding area of research. Papers like [92, 82, 58, 83, 44, 34, 60, 1] have mainly leveraged language for communicating goals or instructions to a robot to plan around or execute, often using language models as intuitive priors over appropriate actions, rewards, or dynamics. For scene understanding, [35] use occupancy maps with language embeddings attached, while [42, 79] connect such information to neural radiance fields. These works focus on lower-level semantics, compared to the more abstract semantics considered here. Past approaches for room classification use an explicit Bayesian probabilistic framework

for determining room labels based on detected objects [16]. However, such methods remain hard to generalize to new rooms and objects. To address these shortcomings, we explore the ability for large language models to be used as common-sense mechanisms for robot scene understanding.

2.2 Language-only Methods

Modern language models (LMs) are generally transformers [93] that learn distributions over strings. We write

$$\Lambda(W) \approx \log p(W) \tag{2.1}$$

to be a LM's estimated log probability score for string W. Recently, *decoder-only causal language models* have seen particular success. Said LMs decompose the above probability via the chain rule such that each subsequent token is conditioned on the ones before it:

$$\Lambda(W) = \Lambda(w_1) + \sum_{i=2}^{|W|} \Lambda(w_i \mid w_{1:i-1}) \approx \log p(w_1) + \sum_{i=2}^{|W|} \log p(w_i \mid w_{1:i-1}) \tag{2.2}$$

where w_i is the i-th token of W.

LMs are also typically able to embed texts into semantically-meaningful vectors, which can be useful for fine-tuning in down-stream tasks. When pre-trained on large data corpora, all these models' outputs can reflect common world knowledge [49]. We thus make use of these capabilites for scene understanding.

2.2.1 Query Strings and Informativeness

If we want to use LMs to reason about a room's class, we first need to project its semantic features (e.g., contained objects) into language. We thus start by building query strings describing a given room whose label we wish to infer. For this, we assume access to a list of objects within. This list can be inferred by existing mapping techniques [76, 36]. For our experiments, we use ground truth object labels from our considered dataset.

Putting *all* objects in a room into the query may result in poor performance, as the

queries may be dominated by uninformative, ubiquitous objects (e.g., lights). We thus draw from Grice's conversational maxims and rational speech acts [28, 27]. In this framework, a pragmatic speaker chooses an utterance based on how a listener would interpret a literal speaker. E.g., a literal speaker might describe a bedroom as "containing chairs," which is true, but ambiguous to a listener who is trying to discern the room's label. A pragmatic speaker would thus instead say the room contains a *bed*, which is much more informative and disambiguating.

To implement this in our queries, we only include the k objects most informative for room classification, noting that *objects which appear in fewer room types are more informative*, as their presence heavily implies certain rooms. Quantitatively, these objects have more *non*-uniform distributions $p(r \mid o)$, where $o \in L_O$ is the object label, $r \in L_R$ is the room label, and $L_{O,R}$ are the sets of all possible objects/rooms respectively. We compute these conditional probabilities in two ways:

- **Using ground-truth co-occurrences**. When using these empirical conditionals, we apply Laplace smoothing. We note this requires task-specific data.
- **Using proxy co-occurrence probabilities by querying language models.** Specifically:

$$p(r \mid o) \approx \frac{\exp \Lambda(W_{o,r})}{\sum_{r' \in L_R} \exp \Lambda(W_{o,r})} \tag{2.3}$$

  where $W_{o,r}$ is the query string "A room containing o is called a(n) r."

With $p(r \mid o)$ available, a natural measure of its non-uniformity (and thus informativeness) is entropy:

$$H_o = -\sum_{r \in L_R} p(r \mid o) \log p(r \mid o) \tag{2.4}$$

Entropy is maximized when $p(r \mid o)$ is uniform and minimized when one-hot, so more informative objects have *lower* H_o. Thus, to pick objects for the queries, we take the k different lowest-entropy present objects:

$$O_{\text{best}} = \underset{o \in O}{\text{argmin\_}k} \; [H_o] \tag{2.5}$$

where O is the set of all object labels contained within the considered room. We thus now

have a set of k objects O_{best} which can be used to infer the room label.

2.2.2 Zero-shot Approach

For the zero-shot approach, we construct $|L_R|$ query strings, one per room label:

$$\begin{aligned} W_r = \text{“A room containing } o_1, o_2, \text{ ... and } o_k \\ \text{is called a(n) } r.\text{”} \ \forall r \in L_R \end{aligned} \tag{2.6}$$

where $o_{1...k} \in O_{\text{best}}$ are ordered by ascending entropy. All these queries are scored via language model, with the final estimated room label $\hat{r}$ being whichever one yields the highest query sentence probability:

$$\hat{r} = \arg\max_r \Lambda(W_r) \tag{2.7}$$

We note that this can similarly be done by prompting the LM to *generate* its inferred room type. However, for evaluative purposes, we follow works like [1] and opt for this scoring approach to constrain the LM to our considered labels.

2.2.3 Embedding-based Approach

For our embedding-based fine-tuning approach, we create a query of the form:

$$W = \text{“This room contains } o_1, o_2, \text{ ... and } o_k.\text{”} \tag{2.8}$$

This string is then fed into a LM to produce a summary embedding vector. Finally, the embedding is fed into a trained classifier, which produces $|L_R|$ prediction logits corresponding to the room labels, with the inferred room label corresponding to the maximum logit. We choose this network to be a shallow multi-layer perceptron.

2.2.4 Structured-data Approach

The above approaches limit LM inputs to natural language queries. This is restrictive, as robot perception systems tend to detect spatial features, like room size, object poses, and

bounding boxes, that are not commonly expressed in natural language. Humans tend to describe spatial relations qualitatively (e.g., "to the left of" or "on top of"), so numeric spatial features may not be processed well via pre-trained LMs. These features are nonetheless useful for room classification: e.g., hallways are often characterized by being *long*.

We thus consider a *structured-data* approach wherein an entire LM is fine-tuned to accept description strings that can easily express spatial features, but are not in the realm of common natural language. This way, the inputs are more expressive, but we still capitalize on the LM's common-sense semantic understanding when linking objects to associated rooms. We use an encoder-decoder LM to encode structured data strings containing the room axis-aligned dimensions and each contained object's label, count, and position (relative to the center of the room):

Room Size:
x *[x room bounding box length]*
y *[y room bounding box length]*
z *[z room bounding box length]*

Object Locations:
[object 1 label]
x *[x position relative to room center]*
y *[y position relative to room center]*
z *[z position relative to room center]*

[object 2 label]
x *[x position relative to room center]*
y *[y position relative to room center]*
z *[z position relative to room center]*

(repeat for all other objects in the room)

The LM's decoder then computes probabilities for each possible room label string conditioned on the structured description, with the highest-scoring one being the inferred label.

2.3 Vision-and-Language Methods

When dealing with grounded tasks, LMs require observations be projected into string space. Some features are invariably lost in this process. In our case, we only use object labels and spatial features, while visual features like color or texture go unused. Such data is still useful: e.g., bathrooms often have white tiles that other rooms do not have, so detecting such features aids in room classification.

We thus look to visual language models (VLMs): pretrained models that connect text and language. For instance, [72] uses a contrastive objective to learn a shared embedding space for vision and language, allowing one to match images with text. Other approaches generate text conditioned on input images and text prompts, usually for visual question answering (VQA) or captioning [51, 52]. We try to use such models to classify locations based on previously-available object information *and* egocentric room images.

2.3.1 Zero-shot Approach

We consider a VLM equivalent to the language-only zero-shot approach. For contrastive VLMs, we follow the standard zero-shot classification technique [72] and embed both room label strings and images from the considered room. The inferred label is whichever room string best matches the images in embedding space (as determined by cosine distance).

For generative VLMs, we input the images and a prompt ("What type of room is this?" for VQA models and "This room is a(n) *r*." for captioners). Such prompts can be prepended with object descriptions – an advantage they have over the contrastive VLMs. The model then follows up the prompt with each possible room label, with the highest-probability label being the inferred one.

2.3.2 Fine-tuning Approach

This approach is the same as the zero-shot VLM approach, but with the model fine-tuned to output room labels. We train a captioning VLM to correctly generate the room *r* in the prompt sentence "This room is a(n) *r*." when given images of said room. Again, at inference

time, the inferred room label is whichever one yields the highest probability when plugged into the end of the prompt.

Note that, for both these approaches, we use VLMs that perform captioning/VQA on a single image at a time. We detail how to get an overall room classification from many images in Section 2.4.4. Our approach can be easily generalized to VLMs that accept multiple input images, like [2]. Due to computation limits, we leave such experiments to future work.

2.4 Experimental Setup

2.4.1 Datasets

We evaluate our algorithms on scene graphs produced from the Matterport3D dataset [15], which is commonly used in robot navigation tasks [59, 98, 90]. This dataset can rapidly render rooms and contained objects, each with labels, pose, and bounding boxes that we use to create scene graphs. Objects are assigned labels from two label spaces: mpcat40 (35 labels) and nyuClass (201 labels) [84]. In total, there are $\sim$ 1870 rooms, each with one of 23 room labels. See Appendix A.1 for scene graph construction details. We now consider how datasets are constructed for pure-language and vision-language approaches respectively.

Language-only: We divide the buildings into a 50/20/30 train/validation/test split for each label space. To produce queries for our embedding-based approach, we bootstrap subsets of the most informative objects per room to put into the query. We generate four such datasets by varying object label space (nyuClass/mpcat40) and co-occurrences used for object selection (ground truth/proxy). We use RoBERTa-large as our LM embedder [54]. See Appendix A.2 for more details. For the structured-language dataset, for each room, we create structured language strings summarizing the room dimensions and object positions and labels (for a given label space). Unlike the zero-shot/embedding approaches, said strings contain *all* objects in a given room. See Appendix A.3 for some additional minor details.

Each dataset for a certain label space is produced from the same splits. All approaches and baselines are tested on the same test split. For completeness, approaches that do not require training are also evaluated on the entire dataset.

Vision-and-Language Datasets: For each room, we sample and save images from 100 random freespace camera poses (see Appendix A.4). For methods that require training, we divide the rooms into a 40/20/40 train/validation/test split. Note that (i) all images for a given room belong to the same group and (ii) this is a different divide than for language-only, as it includes rooms with no objects (as visual features can still be extracted from them).

2.4.2 Baselines

We consider three baselines. First, we use ground-truth co-occurrence data for a *statistical baseline*. We approximate the probability of a room label as the product of the conditional probabilities of the room given each object individually:

$$p(r \mid O) \approx \prod_{o_i \in O} p(r \mid o_i) \tag{2.9}$$

where the conditionals $p(r \mid o_i)$ are empirically estimated from the dataset. The inferred label is thus $\arg\max_r p(r \mid O)$. Second, we train a *GraphSage graph neural network baseline* [31] to predict rooms given objects using the language-only dataset splits.

Finally, as a baseline for vision-based methods, we use the vision-and-language dataset splits to train a ResNet-50 [33] model to predict room label logits given images from the room. We do this from scratch and starting from the pre-trained weights. See Appendix A.5 for details.

2.4.3 Language-only Trial Specifications

Zero-shot: We vary the ground truth/proxy object-room co-occurrences (when computing object entropy) and the object label spaces for four total conditions. We choose $k = 3$ objects per room to create the corresponding queries (or all if the room contains fewer than

three objects). For all trials, we use the GPT-J language model [95] for both inference and generating proxy co-occurences. Due to hardware limitations, we use the half-precision release of the model.

Embedding-based: We train head networks on each of the four embedding datasets. See Appendix A.5 for more details. We also run two generalization experiments. First, we train the network while holding out all nyuClass rooms whose query strings contain chairs, sinks, toilets, beds, and washing machines, then we test on these held out datapoints. This is done with ground-truth co-occurrences. Second, we train models on mpcat40 data while *testing* on nyuClass data in order to see if they can accommodate and generalize to a different, larger input label space. In this case, we divide the mpcat40 dataset using a 40/60 training/validation split, use the entire nyuClass dataset for testing, and vary the co-occurrence type (ground truth and proxy).

Structured-language: We fine-tune a pre-trained T5 LM to encode structured-language strings and decode (score) room labels [73]. We train for 5 epochs and test on the weights with highest validation accuracy. See Appendix A.5 for more details. We also run some ablation trials, removing the room bounding box dimensions, object positions, or both.

2.4.4 Vision-and-Language Trial Specifications

Zero-shot: We test CLIP, BLIP, and BLIP-2's zero-shot visual room classification abilities [72, 51, 52]. Of these models, CLIP is contrastive while the BLIP models are generative, so we adopt the VQA and captioning approaches detailed in Sec. 2.3.1 respectively.

As each room has many images, we measure both the portion of individual images and overall rooms classified correctly as image-wise and room-wise accuracies, respectively. A room is classified correctly if, when the scores for each label are summed over all images from said room, the correct label has the highest total score. This is *different* than a plurality vote system wherein the most-predicted room label (out of all image predictions) is the room label, as our method incorporates model uncertainties by using the output logits/scores.

Fine-tuning: We fine-tune the BLIP VQA VLM to decode room labels based on im-

ages and descriptions [51]. The setup is the same as the zero-shot case, but with the introduction of fine-tuning. The model itself is an encoder-decoder VLM that performs bi-directional self-attention on the question prompt in conjunction with cross-attention on input image features before decoding an answer with causal self-attention while cross attending to image and question embeddings. See Appendix A.5 for more training details. Again, we measure image- and room-wise accuracies with the same maximum score metric as above.

2.4.5 Additional Trials

Building Trial Specifications: We extend the above language-only formulations to also perform inference of a building's label $\in L_B = \{\text{house}, \text{ office complex}, \text{ spa resort}\}$ based on the rooms within. We consider three approaches: (i) the GraphSage baseline, but trained to predict buildings based on known objects and rooms; (ii) the zero-shot approach with $k = 4$ and using ground-truth co-occurrences; and (iii) the embedding-based approach with output size $|L_B| = 3$ and stochastically bootstrapped training data. We present (ii) and (iii)'s accuracies on the same test split our GNN is tested on for comparison. For completeness, we also present (ii)'s accuracy on the entire Matterport3D dataset. For more details, see Appendix A.6.

Real Scene Graph Trial Specifications: We run the zero-shot approach to label rooms on dynamic scene graphs generated using the Hydra spatial perception system [36]. We consider three environments: a dormitory, an apartment, and an office. The object labels are noisily inferred using a fine-tuned HRNet [96] that classifies household objects into 23 labels. We consider six room labels (kitchen, bathroom, hallway, lounge, stairwell, and bedroom) while using proxy co-occurrencies to pick query objects. All rooms with at least one object are evaluated.

2.5 Results

We present all results for language-only and vision-and-language (as well as associated baselines) trials in Tables 2.1 and 2.2 respectively, with parenthesized values indicating

	Baselines		Zero-shot	
	Statistical	**GraphSage**	**GT**	**Proxy**
nyuClass	57.7% (50.6%)	64.2%	52.2% (53.6%)	27.2% (28.2%)
mpcat40	44.7% (46.9%)	59.1%	48.9% (50.1%)	27.5% (27.3%)

	Embedding-based		Structured-language			
	RoBERTa		With Position		No Position	
	GT	**Proxy**	**With Bbox**	**No Bbox**	**With Bbox**	**No Bbox**
nyuClass	65.5%	57.6%	**69.1%**	68.7%	67.3%	67.7%
mpcat40	63.9%	58.5%	**68.3%**	67.7%	63.9%	63.7%

Table 2.1: Test-split accuracies for all language-only approaches. Methods that do not require training have full dataset accuracy reported in parentheses. Structured-language yields the highest test split accuracies (bolded).

evaluation accuracy on the entire dataset (for approaches that do not require training).

2.5.1 Language-only Results

Zero-shot: As shown in the top half of Table 2.1, the zero-shot trials yield room classification accuracies of $27.3 - 52.6\%$ when run on the entire dataset. The ground-truth co-occurrence trials perform better than the statistical baseline evaluated on the whole dataset, which also uses ground truth frequencies. No trial outperforms the GraphSage baseline, but said baseline requires training and cannot be easily extended to additional labels, unlike our approach; by virtue of being zero-shot, the only step needed to adopt the large label space was to compute the informativeness metric for the new objects. This approach also achieves high accuracies for several common household rooms (bathroom, bedroom, kitchen, and living room). For the best performing trial, accuracies for these key rooms range from $79.2 - 97.1\%$ (see Figure 2-2). There also are two trends for when a room will *not* be classified correctly:

- **Room lacks disambiguating objects:** Bathrooms and bedrooms have objects almost exclusive to them (e.g., toilets and beds), but rooms like lobbies and family rooms only contain more ubiquitous ones (e.g., tables and chairs), and so are harder to identify.

	Zero-shot						
	CLIP	**BLIP VQA**			**BLIP-2 Captioner**		
	None	**None**	**nyuClass**	**mpcat40**	**None**	**nyuClass**	**mpcat40**
Room-wise	36.5% (32.7%)	37.8% (36.2%)	37.0% (36.6%)	37.1% (37.5%)	47.5% (45.7%)	47.9% (45.7%)	48.0% (46.2%)
Image-wise	26.5% (25.9%)	30.1% (28.8%)	35.6% (34.9%)	34.4% (34.3%)	40.1% (39/3%)	45.0% (43.1%)	44.5% (43.0%)
	Baseline				**Fine-tuning**		
	ResNet-50				**BLIP VQA**		
	From Pretrained	**From Scratch**			**None**	**nyuClass**	**mpcat40**
Room-wise	51.1%	26.2%			53.2%	**68.6%**	65.3%
Image-wise	36.8%	22.6%			47.0%	**67.9%**	64.1%

Table 2.2: Test-split room- and image-wise accuracies for all language and vision approaches. Methods that do not require training have full dataset accuracy reported in parentheses.

- **Room is not "standard":** Bars, libraries, and spas all more commonly refer to *buildings*, not rooms. We note this could likely be fixed with further prompt tuning and few-shot examples.

For the other rooms, the language model shows the desired common sense when classifying them. Our approach also demonstrates generalization, handling the smaller, 35-object label space (mpcat40) *and* the much larger, 201-object label space (nyuClass). In fact, the nyuClass trials result in higher accuracies than their mpcat40 counterparts, as nyuClass's labels are more specific and informative. This benefit is best shown in the following cases:

- **Kitchens & laundry rooms:** Both rooms are characterized by appliances. While nyuClass provides fine-grained labels (e.g., washing machine vs. stoves), mpcat40 groups all those objects under the broad and ambiguous category of "appliances," making differentiation of the two room labels difficult.
- **Game rooms & garages:** Game rooms are characterized by recreational objects, like ping-pong/foosball tables. Both appear in nyuClass, but are just "tables" in mpcat40, making these rooms easier to identify with the former. Likewise, garage doors (nyuClass) are just called "doors" in mpcat40.

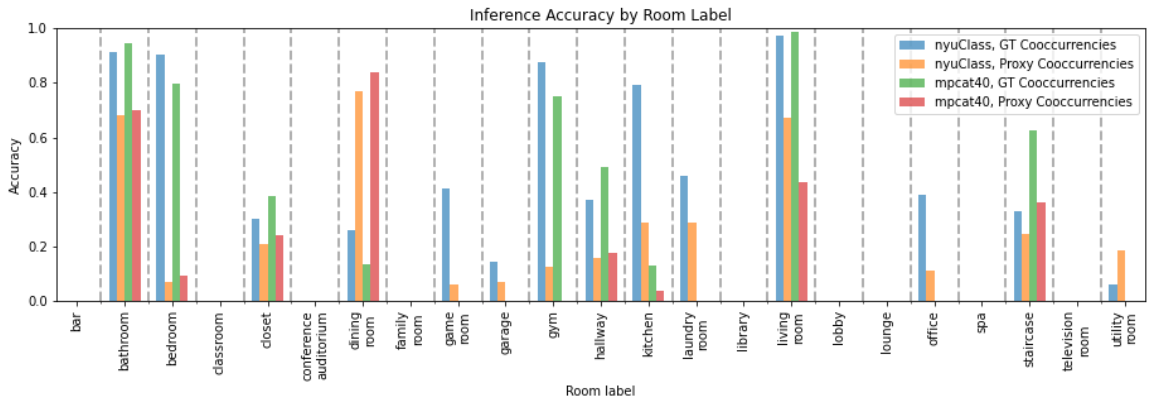

Figure 2-2: Zero-shot accuracies on all data for all conditions, by room label.

Finally, all trials take ~ 1.2 seconds to infer a room's label, with most of the overhead being language model evaluations.

Embedding-based: The embedding-based approach achieves $57.6 - 65.4\%$ test accuracy, beating the statistical baseline ($46.9 - 52.7\%$) and the zero-shot approach ($27.2 - 52.2\%$) in all conditions (see bottom left of Table 2.1). Notably, unlike for the zero-shot case, this is even true for the conditions using proxy co-occurrences, so the model does not explicitly need ground-truth co-occurrences for picking out objects to achieve high performance (though some is learned in training). Trials using ground-truth co-occurrences yield slightly higher accuracies than corresponding GraphSage baselines, while the proxy trials are still competitive. Finally, RoBERTa has $20\times$ fewer parameters than GPT-J (used for zero-shot), meaning this approach requires less compute, memory, and time to run inference.

Holdout Trial Room Accuracies				
Washing Machine	**Chair**	**Sink**	**Toilet**	**Bed**
14.6%	40.0%	64.2%	77.6%	83.5%

Table 2.3: Test accuracies for rooms containing each holdout object.

Embedding-based Transfer: Using language models to embed the query room descriptions enables some generalization to novel object classes. The approach does well for some objects in the holdout object trials (see Table 2.3). Even when held out in training, it correctly classifies rooms whose queries contain sinks, toilets, and beds $64.2 - 83.5\%$ of

the time. The networks likely learns to extract essential information on room labels from query embeddings containing only non-held out objects that generalizes to the held out ones. E.g., while toilets are held out, related objects like bathtubs are not. Due to their relatedness, the embeddings for observed queries ("This room contains bathtubs.") may be similar to that of unobserved ones ("This room contains toilets."), so the network classifies the latter correctly too. As there are no objects related to washing machines in nyuClass, rooms containing them have comparatively low accuracy. Still, as most rooms have several characteristic objects, holding out a subset of them (or, equivalently, introducing more of them) is generally not an issue.

Regardless, our approach shows promising generalization and transferability. When trained on mpcat40 and tested on nyuClass, it yields 47.1% and 56.6% accuracy for proxy and ground-truth co-occurrences respectively, comparable to the best accuracy for the zero-shot approach (52.57%) and some non-transfer fine-tuning conditions.

Structured Language: We intuitively expect the structured-language approach to do as well or better than the embedding approach, as the information contained within the structured language string should be a superset of that of the embedding-based inputs. Sure enough, all structured-language approaches have test accuracies from $63.69-69.12\%$, being comparable to or better than all other approaches (see bottom right of Table 2.1). We note that ablating room bounding box information does not degrade performance much, but removing object positions is more impactful (especially for mpcat40 trials). Nevertheless, all changes from ablations are relatively small; generally, *structured language of any type seems to be the best overall language-only approach for room classification*.

2.5.2 Vision-and-Language Results

Zero-shot: The BLIP-2 trials have the highest zero-shot room- and image-wise accuracies, comparable to that of the fine-tuned ResNet-50 baseline, despite having no task-specific training (see top of Table 2.2). Adding object labels (from either label space) to the prompt improves image-wise accuracy, likely because it aids in the classification of uninformative images (e.g., ones that face walls) by providing non-visible room-wise information on

contained objects. However, it surprisingly does *not* improve room-wise accuracy, perhaps because the improved image classifications are already for rooms that are visually distinct and thus can be classified from just the good images.

We find that the BLIP-2 approach has slightly lower accuracy than the best language-only zero-shot trial ($45.66\% - 46.16\%$ vs. 52.57%), though very slightly higher than the second-best trial. However, the former requires (i) less prompt engineering and (ii) no ground-truth co-occurrence data (which is needed to pick objects for the query template in the language-only case), so is overall easier to implement and use.

Fine-tuning: All vision approaches that involve training/fine-tuning are presented at the bottom of Table 2.2. The fine-tuning approach yields similar accuracies to the structured-language approach. Notably, in contrast to the zero-shot equivalents, the fine-tuned models' image- *and room-wise* accuracies benefit from object labels in the prompt. This suggests that, unlike with zero-shot, the fine-tuned models both use the object labels to correctly classify aforementioned uninformative images *specifically for rooms that would otherwise not be correctly predicted* – it is not just correcting uninformative images in rooms whose other images are easily correctly classifiable.

Moreover, we see that the fine-tuned trial that does not have object labels achieves a room-wise accuracy of 53.16%, compared to the pre-trained ResNet-50 vision-only baseline accuracy of 51.05%. These values are similar and both higher than the from-scratch ResNet-50 baseline trial, suggesting that pre-training (be it visual or visual-linguistic) aids in transfer to the room classification domain.

However, both pre-trained ResNet-50 and the fine-tuned no-label trials fail to beat *any* of the approaches that use fine-tuning in conjunction with object labels (including the language-only approaches). This suggests that, even when trained on domain-specific data, it is difficult for vision alone to classify rooms from arbitrary images within.

It is thus appropriate to supplement such inferences with room-wise object information, especially if such data is already available via a robot metric-semantic spatial perception system. In such a case, we find that (visual) language models can easily incorporate said object labels into their subsequent abstract room label inferences, leveraging their pre-trained common-sense understanding of such object-room relations to improve room classification

accuracy.

2.5.3 Building Trial Results

Building Labels	Statistical Baseline	GraphSage Baseline	Zero-shot Approach	Embedding-based Approach	Total in Dataset
House	27	26.6	25	25	27
Office Complex	0	0.4	2	2	2
Spa Resort	0	0	1	0	4
Total	27 (81.8%)	27 (81.8%)	28 (84.8%)	27 (81.8%)	33

Table 2.4: Test accuracies of building inference approaches.

All building-labeling approaches yield similar test accuracies (Table 2.5.3). The statistical baseline only identifies houses correctly, as the dataset (and thus co-occurrencies) is dominated by houses. The zero-shot and embedding-based approaches successfully classify both office complexes, while GraphSage identifies only 0.4 on average over five trials. All approaches do not identify spa resorts well. For our approaches, we suspect this is due to spa resorts being a non-standard building (the term often refers to entire campuses) and information on resorts' contained rooms (other than spas) not being commonly found in text corpora datasets. This shows a flaw with language models: while humans can reason that resorts contain rooms like offices and lobbies, texts will not usually describe these second-order facts explicitly.

When evaluated on the entire dataset, the zero-shot approach labels $70/81$ (86.41%) buildings correctly, again mainly failing with resorts. While future work *should consider more robust building-labeling benchmarks*, our results show that our language-leveraging approaches can generalize to scene understanding tasks beyond room labeling.

2.5.4 Real 3D Scene Graph Results

The zero-shot approach succeeds at labeling rooms on real 3D scene graphs. However, like on the Matterport3D data, it tends to work best on rooms that had disambiguating objects (bedrooms, stairwells, and kitchens), though the presence of those objects in rooms they

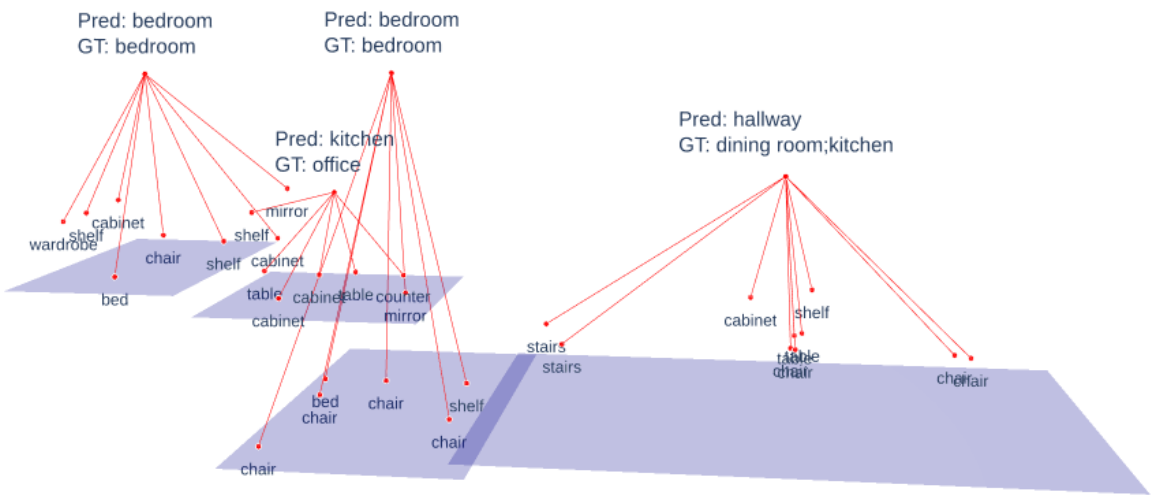

Figure 2-3: Inferred and actual room labels on a real 3D scene graph created by Hydra [36]. Purple regions show the bottoms of room bounding boxes.

are typically not found in also throws off predictions. This is shown in Figure 2-3, where bedrooms are identified correctly but the kitchen and office are not (due to the lack of appliances and the presence of counters respectively). Interactive visualizations and inferred labels for all environments can be found at `https://github.com/MIT-SPARK/llm_scene_understanding#real-scene-graph-labeling-visualization`.

2.6 Conclusion

We show the applicability of large pre-trained LMs and VLMs to the problem of abstract robot scene understanding, particularly in the domain of room classification. We explore an array of different approaches to this problem: zero-shot, embedding-based, and structured-language for language-only and zero-shot and fine-tuning for vision and language. We compare such methods with standard statistical and learned baseline approaches, showing that using LMs yields higher overall performance while also having good generalization to held-out objects and transferability to new label spaces. Our results show that these paradigms are promising avenues of development for scalable, sample-efficient, and generalizable robot spatial perception systems.

Limitations

Our primary limitation is compute. We ran all language model inference experiments on a single RTX 3080 GPU, which vastly constrains the size of model (and thus the resulting experiments) we could use. In particular, while not quantitatively reported, we find that the zero-shot approach empirically benefits greatly from using larger models trained on more data; smaller equivalent models like GPT-Neo [9] or GPT-2 [71] yield very poor zero-shot results. We found that the six billion parameter GPT-J was the largest such model we could reliably load, but still had to use the half-precision release due to memory constraints. Likewise, our structured language approach only includes room dimensions, object labels, and object positions; some spatial/geometric information, like object pose, is still lost, as they do not fit in our smaller models' token counts (or, even if they do, would take up too much memory).

We thus detail some additional research directions that would be possible with larger compute budgets. We suspect that newer, larger models (on the scale of $10 - 100$ billion parameters) would improve performance even more, especially when combined with recent advances in instruction and human preference-based tuning. For the VLM experiments, newer models explictly trained for embodied visual-linguistic reasoning [24] would also likely perform very well for our scene understanding tasks. Finally, many common-sense visual understanding problems have shown to work well with Socratic model [101] setups, wherein multiple models (typically covering multiple modalities) interface with each other in natural language.

However, these constraints *do* also reflect a real challenge if such systems are deployed on real mobile robots – such autonomous systems generally do not have the hardware required to load the latest and largest language models on them locally. It is therefore promising that, even with somewhat older, smaller models (which *could* feasibly be deployed on a robotic platform), we still achieve relatively good performance in our considered tasks.

It thus would be exciting to see both how one could get the most out of the limited compute of a mobile robot *and* how our scene understanding approaches could be made more effective when the limitation is relaxed.

Ethics Statement

When applying large pre-trained models to robotics, it is very important to recognize how the risks associated with such models can be amplified when given embodiment [11]. We focus on the relatively innocuous use case of scene understanding/abstract location classification, but even still, the biases captured within our considered models may be reflected in their performance on such tasks. For instance, we evaluate on the Matterport3D dataset, which contains scans of domestic environments. It is safe to say this data represents the homes of a narrow intersection of wealth, class, and culture. It is possible that homes of certain demographics may not conform to the prototypical conceptualization of such locations present within our (visual) language models, *and* that our evaluative metrics do not detect such biases (again, as our testing data is not diverse enough).

Thus, we believe it is important for such data to be diverse and transparently/ethically sourced (while maintaining proper privacy), especially when used to train general-purpose household robots. In addition, we encourage research into more advanced scene understanding techniques that allow for interpretability of inferences and adaptability to a wider range of environments.

Chapter 3

Robot Navigation

As shown in Chapter 2, **common-sense priors** are a vital part of performing inference in the real world. Abstract contextual information can help with tasks like visual recognition. For instance, a large white box in a household environment might be an oven or a washing machine. However, if one sees this object accompanied by a refrigerator and sink, then one can easily infer that the room is a kitchen and that the object is an oven. Such prior expectations and top-down understanding are omnipresent in human reasoning for tasks like object recognition (such as in the example) and written text interpretation [45, 63].

Contrastingly, modern machine learning tends to learn from vast amounts of bottom-up raw sensory input *without* the supervision of common sense. Thus, they (i) require specialized simulation environments or large amounts of (often annotated) data and (ii) reflect the biases, correlations, or distibutional quirks of their training datasets, even when undesirable or illogical [10, 46]. When training a model, it remains difficult to expose it to data corresponding to all the scenarios it is likely to encounter in real-world deployment.

As correcting these faults is difficult (if not intractable), we turn to language. There is evidence that humans benefit from linguistic supervision when learning common-sense knowledge that is difficult to glean from experience [70]. This is computationally reflected in the recent successes of large language models in various language processing tasks. Language models have been used as sources of common sense and domain knowledge for question-answering [91], scripting text games/stories [3, 4], and probabilistic programming [48].

With the development of more advanced multi-modal models that use text, these models have also seen success in grounded and embodied tasks. We call one popular approach **model chaining** (MC) [1, 101], as it involves interfacing models with text by projecting observations, task specifications, and decisions into natural language. However, this approach is naturally limited by the difficulty of expressing and using certain features (e.g., numeric quantifications of uncertainty) in the natural language "thoughts" of the model ensemble.

This motivates our current work. We present a method for extracting common-sense knowledge from language models in the form of probabilistic priors, which can then be integrated with probabilistic formulations of inference tasks. As many standard machine learning tasks can be framed as probabilistic graphical inference, this approach offers flexible opportunities for language model knowledge to be inserted – the model can provide priors over labels, decisions, or even parameters. We call this approach **LaMPP**, for **La**nguage **M**odels as **P**robabilistic **P**riors.

While the full paper [49] shows LaMPP's applicability to a range of tasks, for this chapter, we focus on how such a method is useful for robot navigation specifically. In particular, we apply it to a modified version of the Habitat Challenge ObjectNav task [98], which has the task objective of developing an agent to find, identify, and move to a specified target object based on visual sensor readings as quickly as possible in an unfamiliar simulated household environment (drawn from the Habitat dataset [74, 59, 90], just as the experiments in Chapter 2).

In Section 3.2, we present LaMPP as a general framework for querying language models for relevant probabilistic priors and inserting those outputs into domain models. Then, in Section 3.3, we see how LaMPP can be used for robot navigation in the previously-mentioned task. Lastly, Section 3.4 briefly touches upon the insights gained from applying LaMPP in non-robotics contexts – specifically, image semantic segmentation and video action recognition.

Overall, we hope to highlight how language can be a valuable source of common sense for robot control and decision-making, as opposed to just inference (as shown in both our discussions on scene understanding and in the other domains where we apply LaMPP). Moreover, we show the importance of composing linguistic information with existing prob-

abilistic inference techniques in a principled manner, as opposed to techniques that purely work in string space (which may lack inductive biases critical to considered tasks).

3.1 Related Works

Model Chaining: There has been much recent work in combining and composing the functionality of various models *entirely in string space*. For instance, the Socratic model framework proposes chaining together models operating over different modalities by converting outputs from each into text [101]. Zero-shot inter-model interactions are then entirely facilitated by natural language. For example, for image captioning, a visual language model might detect high-level features (locations, objects, etc) which are inserted in a template prompt that a generative language model completes with possible captions. These can then be fed back into visual language models to see which text is best grounded in the image.

While these methods have yielded good results in a variety of tasks, like egocentric perception and robot manipulation [1, 85], they are fundamentally limited by the expressivity of their string-space interfaces. Models often output useful features that cannot be easily expressed in language, such as probabilistic uncertainty (like in a traditional image classifier). Even if such information is written in string form, there is no guarantee that language models will correctly use it for formal symbolic reasoning – e.g., if a language model is told $p(A) = 0.1$ and $p(B \mid A) = 0.5$ in an input string, when prompted for $p(A,B)$, it often will not output the correct value of $0.1 \times 0.5 = 0.05$ [99]. This motivates our use of language models *not* as an oracle decision-maker, as [1, 85] do, but as explicit probabilistic priors that can be combined with domain models that operate over or output relevant probabilities.

Concurrent to the present work is [18]. While they similarly uses language model scores as a source of common-sense prior information for other decision-making tasks (feature selection, reward shaping, and causal inference), they likewise do not make use of explicit *priors for probabilistic models*.

Language Models and Probabilistic Graphical Models: There is a long history of language modeling itself being interpreted as probabilistic graphical inference [81]. How-

ever, recently, more abstract tasks that leverage language models have also been viewed in this way. Methods like chain-of-thought question-answering [97], thought verification [20], and bootstrapped rationale-generation [100] may all be interpreted as probabilistic programs encoded as repeated language model queries, dubbed "language model cascades" [22]. However, this analysis exclusively considers language tasks; to the best of our knowledge, this is the first work to specifically connect language model evaluations to probabilistic graphical models in non-language domains, including for robotics.

3.2 Methods

We consider two broad classes of problem formulation, which many machine learning tasks can be interpreted as. A basic formulation is presented as follows.

First, in **prediction** tasks, we try to model $p(y \mid x)$, where y is some set of labels and x is an observation. This can be factorized as:

$$p(y \mid x) \propto p(y) \cdot p(x \mid y) \tag{3.1}$$

where $p(y)$ is a prior over labels (which can be learned or extracted from an external source – in this case, language models) and $p(x \mid y)$ is a generative observation model. This equation provides a simple way of combining priors with domain-specific pretrained models' outputs.

Note that, oftentimes, tasks cannot be framed so easily. It often helps to also introduce *auxiliary* latent variables y', which can then be related to labels y and observations x. This is done in our navigation task, as discussed in the next section.

Second, in **learning** tasks, we have some generative probabilistic graphical model whose parameters θ are (i) interpretable and (ii) learned from data $\mathscr{D}$. For example, in our video action recognition task, we consider a hidden Markov model (HMM) as one such case. The task this time is to find the parameters that best fit the data, $\arg\max_\theta p(\theta \mid \mathscr{D})$,

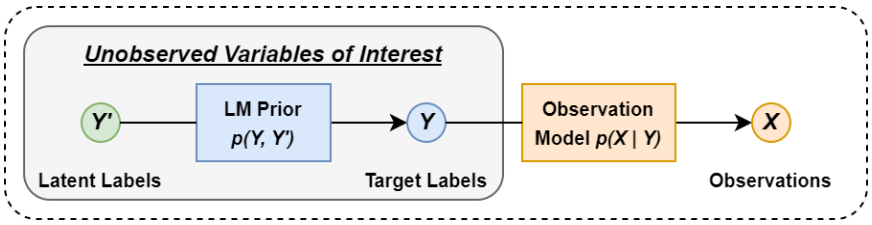

Figure 3-1: A general illustration of how to apply LaMPP to the probabilistic generative framing of a task. The language model provides probabilistic priors connecting auxiliary and target variables.

which factorizes in the following way:

$$
\begin{aligned}
p(\theta \mid \mathscr{D}) &\propto p(\mathscr{D} \mid \theta) \cdot p(\theta) \\
\Rightarrow \arg\max_{\theta} p(\theta \mid \mathscr{D}) &= \arg\max_{\theta} p(\mathscr{D} \mid \theta) \cdot p(\theta) \\
&= \arg\max_{\theta} \prod_{\{x,y\} \sim \mathscr{D}} [p(x, y \mid \theta)] \cdot p(\theta) \\
&= \arg\max_{\theta} \prod_{\{x,y\} \sim \mathscr{D}} [p(y \mid x, \theta) \cdot p(x \mid \theta)] \cdot p(\theta) \\
&= \arg\max_{\theta} \prod_{\{x,y\} \sim \mathscr{D}} [p(y \mid x, \theta)] \cdot p(\theta)
\end{aligned}
\tag{3.2}
$$

where $p(\theta)$ is a prior over parameters (e.g., contributed via regularization). For the last line in this derivation, we assume the model parameters have no effect on the distribution of input data (i.e., $p(x \mid \theta) = p(x)$) and that we have a uniform prior over dataset inputs (i.e., $p(x)$ is constant). As with the prediction tasks, it is often helpful to introduce non-learned hyperparameters α which can be related to θ (which may otherwise be learned).

With these two broad problem structures in place, we thus provide a general step-by-step guide to applying LaMPP. Refer to Figure 3-1 for an illustration.

1. **Choose a domain model** of $p(x \mid y)$ or $p(y \mid x; \theta)$. For instance, the former could be a pre-trained image classification or semantic segmentation neural network.

2. **Design a label space**, which could just be the labels/parameters of interest *or* any auxiliary latent variables that can be related to observations/values of interest, as discussed above.

3. **Query the language model** for probabilistic scores on relations between variables by expressing said relations in natural language. We use GPT-3 as our scorer for all experiments [13].

4. **Perform inference** by using the scores in conjunction with the base model with any appropriate probabilistic compositions.

In Sections 3.3 and 3.4, we show how this framework behaves in three tasks, focusing in particular on the navigation task. These tasks also have different generalization conditions:

1. **Zero-shot**: $p(x \mid y)$ is known and encoded in the domain-specific observation model, but there is no explicit information on $p(y)$ without LaMPP.

2. **Out-of-distribution**: The dataset is biased/distributionally shifted compared to the domain model, with some cases being over-/under-represented. This can be manually induced with adverserial data splits.

3. **In-distribution**: The test and training distributions are the same, but the model is still deficient for some predictive purposes (e.g., on labeling rare classes).

The types of generalization condition considered for each task depends largely on the task specifics (e.g., what data is available, ease of manipulating data splits, what types of existing pretrained models are available). Navigation focuses primarily on zero-shot generalization, as discussed next.

3.3 LaMPP for Navigation

3.3.1 Problem Statement

As mentioned prior, we consider the ObjectNav task in the Habitat simulation environment [98]. In this task, the agent must find the nearest instance of a specified object in an unfamiliar household environment in as few steps as possible. The agent receives virtual egocentric RGBD camera images and 2D position[1]/compass readings (i.e., relative horizontal displacement and yaw to the starting pose). With these observations, the agent must

[1]Referred to in ObjectNav literature as GPS readings.

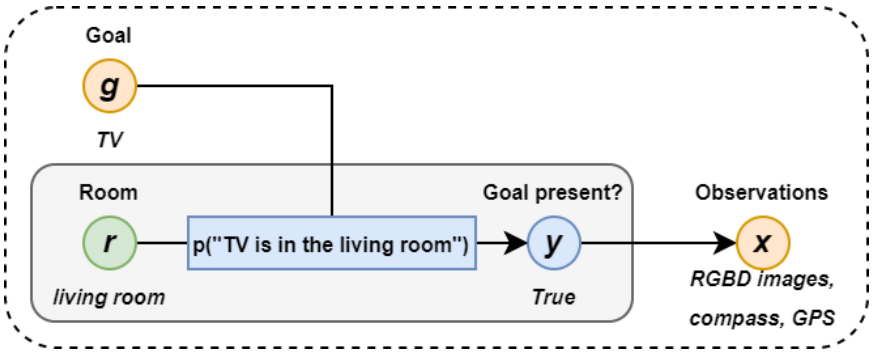

Figure 3-2: The generative structure used in the navigation task. We wish to find locations that maximize the probability of y being true (i.e., the goal object g being nearby) based on observations x and latent room variables r.

operate around a policy with an action space consisting of four commands: *move forward*, *turn left*, *turn right*, and *stop*. The last command should be executed only when the agent thinks it has found the object.

We modify this challenge slightly by providing a high-level map of the environment with room labels. As this information is not available in the version of the Habitat dataset that ObjectNav uses, we manually annotate rooms with labels from a small set of common household locations. This map does *not* include object locations; the agent must rely on its observation model and top-down knowledge of sensible object-room co-occurrences to efficiently and accurately find its goal. Such a modification simplifies the task, but also helps highlight the usefulness of language models at providing top-down priors in the zero-shot regime.

3.3.2 Methods

We now describe how we apply LaMPP to this modified ObjectNav challenge. Note the connections between the following section and the four general steps to our method described in Section 3.2, as well as how the navigation probabilistic structure in Figure 3-2 is a specific instantiation of Figure 3-1.

Base Model. We assume access to a low-level controller that, given a specified (x, y) position, can output actions that move the robot to that location based on the Habitat simulator's provided sensor signals. We specifically choose to use the STUBBORN agent [57].

Normally, this agent would have to construct a map of its environment via exploration, but due to our task modification, it does not do this. In each state, the agent also outputs a scalar score of whether or not the goal object is nearby. Thus, this controller also acts as our noisy observation model.

High-level Policy and Label Space. Our agent has two types of general high-level behavior.

1. **Navigation**: The agent picks a room r to go to, based on how likely the language model judges that room to contain the goal object. Upon reaching that room, the agent explores around for a fixed number of steps.

2. **Selection**: Each time the agent gets an observation, it can also decide to *stop* if it believes the goal object to be nearby.

We define a latent variable y to be an indicator of whether or not the agent is near the desired object g in a given state/location. The task of the navigation step is thus to go to the location that maximizes $p(y = 1 \mid r, g)$ – that is, figure out and go to the room that most likely contains the goal object. In this case, the room is a latent auxiliary variable that relates to the variable of interest g. A full discussion of the probabilistic story for this task is present in [49].

Language Model Queries. This distribution is the prior knowledge that LaMPP fills in for this task, relating latent auxiliary variable r with considered task variable y, all conditioned upon the goal object g. We thus query the language model with the following string:

$$W_{r,g}^{\text{plaus/implaus}} = \text{“A(n) [}r\text{] has a [}g\text{]: [plausible/implausible].”} \tag{3.3}$$

Softmaxing over the scores for the prompts finished by plausible and implausible yields a proxy for the desired distribution encoding prior knowledge, $p(y \mid r, g)$:

$$p(y = 1 \mid r, g) \approx \frac{\exp \Lambda \left(W_{r,g}^{\text{plaus}} \right)}{\exp \Lambda \left(W_{r,g}^{\text{plaus}} \right) + \exp \Lambda \left(W_{r,g}^{\text{implaus}} \right)} \tag{3.4}$$

Where Λ is the log probability estimator function of a language model, as defined in Chap-

ter 2. This can be done for all room and goal object combinations.

Inference. With these priors available, we now define the exact policy used in both high-level behaviors.

1. **Navigation**: The agent chooses the room maximizing $p(y \mid r, g)$, out of the rooms it has not yet visited. As these rooms have not been observed yet, the optimal policy should move to the room most likely to contain the goal *a priori*. The low-level control for this step is done by STUBBORN.

2. **Selection**: The agent chooses to *stop* if $p(y \mid x, r, g) > \tau$, where $\tau \in [0, 1]$ is a thresholding parameter. Note that we can choose the distributional structure of $p(y \mid x, r, g)$. We choose to make a simple independence assumption and decompose it as

$$p(y \mid x, r, g) \approx p(y \mid x) \cdot p(y \mid r, g) \tag{3.5}$$

   i.e., the product of the observation model from STUBBORN and the language model priors.

3.3.3 Experiments

We run the above policy on the ObjectNav test set as the *LaMPP* agent. Additionally, we run three baselines. First, the STUBBORN agent *base model*, which does not use high-level map information at all (and thus we expect to do worse than our agent). Next, we also run a *uniform prior* baseline, which assumes that the object is equally likely to appear in any single room type:

$$p(y = 1 \mid r, g) = \frac{1}{\text{number of room types in environment}} \tag{3.6}$$

Finally, we create a model chaining (*MC*) baseline, taking inspiration from [1, 101]. At a high level, it works by querying the language model for the order it should explore the rooms in, rather than treating object-room relations probabilistically. A full discussion can be found in [49]. For details of the MC baseline, see Appendix B.

Model	Success Rate		Per Class Change in Success Rate	
	Class-Averaged	Frequency-Averaged	Best	Worst
Base Model	52.7%	53.8%	-	
Uniform Prior	52.1%	51.7%	-	
Model Chaining	61.2%	65.3%	Toilet (+20.9%)	TV Monitor (-4.2%)
LaMPP (ours)	66.5%	65.9%	TV Monitor (+33.0%)	Plant (-0.0%)

Table 3.1: Zero-shot navigation results. We report class-averaged (over goals) and frequency-averaged (over episodes) success rates for all baselines and our LaMPP agent. We also report the classes with the best and worst change in success rate for the MC and LaMPP agents, relative to the base model.

3.3.4 Evaluation & Results

Results for all agents are shown in Table 3.1. LaMPP outperforms both the base model and uniform prior baselines, while yielding comparable (but slightly better) results compared to the model chaining case. Notably, LaMPP requires significantly fewer language model queries than MC, as the former generates reusable and precomputable room-object co-occurence probabilities in $p(y \mid r,g)$, whereas the latter needs new queries during each episode.

As a further comparison of the LaMPP and model chaining baseline, we note the importance of the room-object co-occurrence probability in the decomposed selection model. That is, in the navigation step, both LaMPP and MC visit rooms in a similar order – either directly prompting the language model to rate the order in which rooms should be visited *or* generating and sorting probabilistic priors both yield similar results. However, there is a big difference in the *selection* step. The MC baseline effectively exclusively uses the bottom-up observation model's stop selection criteria to determine when it has found goal g, whereas LaMPP needs both the observation model *and* the prior to agree to stop in order to satisfy

$$p(y \mid x,r,g) \approx p(y \mid x) \cdot p(y \mid r,g) > \tau \tag{3.7}$$

Indeed, in additional ablation studies where we remove this part of the LaMPP agent's selection policy, we observe significantly degraded performance compared to the MC baseline. It is thus the integration of top-down language model common-sense priors *with* the

Condition	Model	mIoU	Per Class Change in IoU	
			Best	Worst
In-distribution	**Base Model**	47.8%	-	
	LaMPP	48.3%	Shower Curtain (+18.9%)	Desk (-2.16%)
Out-of-distribution	**Base Model**	33.8%	-	
	LaMPP	34.0%	Nightstand (+8.92%)	Sofa (-2.50%)

Table 3.2: In- and out-of-distribution semantic segmentation results. We report mean intersection-over-union for all conditions and models. We also report the classes with the best and worst change in intersection-over-union relative to the base model for LaMPP.

domain-specific observation model that both yields the best overall performance and the best language model query efficiency.

3.4 Other Applications of LaMPP

We will now cover applications of LaMPP in non-robotics domains. As these tasks are less aligned with this thesis's domain of interest, these discussions shall be comparatively brief. Nevertheless, the insights provided by these case studies are still applicable to robot navigation and scene understanding. Again, full details for both these tasks can be found in [49].

LaMPP for Semantic Segmentation. We examine how LaMPP can improve object segmentation in images of household environments by providing priors on (i) the room type being depicted and (ii) what objects look like other objects (and thus are commonly confused by a standard feed-forward semantic segmentation neural network). We use RedNet [38] as a base model, investigating how LaMPP changes performance on the SUN RGB-D dataset [86]. We evaluate on two experimental conditions: in-distribution, where the base model is trained on a standard SUN RGB-D train split, and out-of-distribution, where the base model is trained on a train split that intentionally holds out a certain common object pair co-occurrence (specifically, images that contain both beds and nightstands). We consider the mean and class-specific intersection-over-union (IoU) of both the base model alone as well as the base model supplemented with LaMPP. Values are reported in Table 3.2.

Condition	Model	Step Recall	
		Class-average	Frequency-averaged
Zero-shot	**Base Model**	44.4%	46.0%
	LaMPP	45.7%	47.9%
Out-of-distribution	**Base Model**	37.6%	40.9%
	LaMPP	38.1%	41.2%

Table 3.3: Zero-shot and out-of-distribution video action segmentation results. We report class- and frequency-averaged recall for all conditions and models.

For the in-distribution condition, we see that LaMPP performs slightly better than the baseline. However, we note the best change in IoU ($+18.9\%$) is dramatically larger than the worst degradation (-2.16%). We see this is because the most-improved object label, shower curtains, are often confused with *regular window curtains* by the base model, *despite having seen shower curtains in the training set*. On the other hand, with LaMPP, the system infers that, due to the presence of things like toilets and sinks, the room is likely a bathroom. Moreover, it now has priors stating that (window) curtains look like shower curtains. Thus, by giving priors on both room label context and objects' visual similarities, LaMPP is able to handily fix this deficiency, resolving the uncertainty present even in models that *are* trained on data reflecting the test distribution.

This shower curtain fix occurs in the out-of-distribution condition too, albeit not as dramatically. Instead, the biggest improvement occurs with nightstands, one of the two items in the holdout set. We see that the base model often misclassifies nightstands as tables or cabinets. However, by inferring that the image is of a bedroom and noting that tables and cabinets look like nightstands, LaMPP is again able to somewhat mitigate this distributional shift's adverse effects.

LaMPP for Video Action Recognition. Lastly, we look at a case wherein LaMPP provides priors over model parameters. We consider the CrossTask dataset [102], which contains instructional videos on various tasks (e.g., make pancakes) that are annotated into actions (e.g., add egg). The objective is to order actions according to a given instructional video for a specified task. For a base model, we consider [25], which trains a hidden Markov model (HMM) to both identify how latent variables (actions) (i) give rise to certain video observations and (ii) are locally ordered. We consider two conditions: zero-shot,

where we assume no action transition information in the training set (each action is equally likely to lead to any other, so the base model must rely on bottom-up information from the observation model with no temporal transition probabilities), and out-of-distribution, where we hold out all training examples containing a certain action transition.

In this case, the HMM learns transition probabilities for each pair of actions (y, y'). Normally, the best such estimate of the probability would be the proportion of the time that said action transition occurs (plus some smoothing term):

$$p(\text{Next action is } y' \mid \text{Current action is } y) = \theta_{y \to y'} = \frac{\#(y \to y') + \alpha_{y \to y'} - 1}{\#(y) + \sum_{y''} \alpha_{y \to y'} - |Y|} \tag{3.8}$$

This imposes a Dirichlet prior on the parameters:

$$p(\theta) = \prod_i \theta_i^{\alpha_i - 1} \tag{3.9}$$

Intutively, larger $\alpha_{y \to y'}$ means a particular transition should be more likely. For the base model, all such α values are set equal – no transition is weighted more *a priori*. LaMPP provides an estimate of these parameter values by querying the language model for a score of how likely it thinks a given transition is.

Following [25], we evaluate *step recall* of both the base model and LaMPP in both conditions. Sure enough, as seen in Table 3.3, LaMPP does slightly better than the base model in both cases, demonstrating that the language model has a good sense of the local order of actions to complete a given task, fixing both curated deficiencies in the training data (for out-of-distribution) *and* nonexistent transition information altogether (for zero-shot).

Chapter 4

Large-scale Spatial Ontology

LaMPP shows that the language modeling objective yields good approximate priors over common-sense facts when applied to strings describing those facts. This allows for the automated extraction of probabilistic priors over large sets of common relations, provided they can be programmatically expressed in natural language.

This is very useful for robot spatial perception tasks that require richer, more abstract semantic information, such as supplementing 3D scene graphs considered in Chapter 2. For instance, as described prior, existing spatial perception systems like [36] attach object-level labels, while generating representations of rooms by dilating occupancy grids to find enclosed areas of free space. However, many robotic applications necessitate spatial mapping and perception in indoor *and* outdoor environments. Such scenes, while less spatially structured than rooms, still have useful abstract structure, generally arranged hierarchically: e.g., a grouping of residential buildings and small stores is likely a neighborhood, which is similarly part of a city or town. More natural environments similarly have this kind of structure: low-level perceivable objects, like water, sand, and trees imply the higher-level label of beach or bay.

We thus consider methods for extending scene-graph-based spatial perception to these kinds of richer outdoor environments. Specifically, we wish to use language models to construct a spatial ontology: a hierarchical organization of objects and locations (collectively referred to as *concepts*) and their spatial relations with each other. Such an organization is *ungrounded* – a concept node in the representation does not refer to a specific instance of

the concept found in the real world. Rather, it describes common-sense relations between these abstract concepts, e.g., where a given concept *might be/is often* found. This can then be used for subsequent learning or inference purposes that are applied to actual grounded scenes. For the purposes of this thesis, we will primarily consider the construction and evaluation of these data structures. While we will briefly discuss how they can be used for downstream tasks, we leave extensive analysis of such methods to future works.

4.1 Related Works

Language and Robot Scene Representations: A majority of works applying language models to robot spatial perception have focused on object-level understanding, albeit with different internal mapping representations. For instance, [35] uses a voxel-based occupancy map wherein cells also have language embeddings attached, while [42, 79] connect such information to neural radiance fields. As in Chapter 2, we consider the use of scene graph representations [5]. [80] connects similar graph representations (albeit lacking hierarchy) to visual-language features for navigation tasks, converting spatially-grounded traversal instructions in language into path search over said graphs.

All these works focus on connecting language to visual observations of the scene. As such, they work with visually-perceivable spatial concepts, as opposed to more abstract ones that require "common sense" to identify. We decide to encode said knowledge (which we extract from pre-trained language models) into an ontology – a type of graph that relates concepts together.

Common-sense Ontologies and Taxonomies: Such tasks are common benchmarks in natural language processing. In particular, several works attempt to construct such relational graphs to both measure how well language models capture such common sense *and* how coherent the understanding is. [17] tries to construct WordNet [62] taxonomic subtrees, using a fine-tuned language model to score possible hypernym relations, then uses a maximum spanning tree algorithm to choose which relations to keep while enforcing a tree structure.

Likewise, [30] measures if language models can create relational graphs of components

of everyday objects. They also consider how (i) how consistent the relations are (e.g., if A is inside B, B should not be inside A) and (ii) how to *enforce* relational consistency via a satisfaction solver. We take inspiration from these works in terms of how we attach weights to/prune our ontology graph and how construction of our ontology graph can be automated while maintaining consistency (see Section 4.5.1).

4.2 Methods

4.2.1 The Spatial Ontology

We consider how to construct a **spatial ontology** of common places and objects. We choose to represent this knowledge as a graph, wherein nodes represent concepts (objects or locations at varying levels of abstraction) and edges represent spatial relations that pairs of concepts often have. To simplify the problem, we only consider the containment relation, wherein a directed edge from A to B means that the former is found in the latter. We also distinguish between two types of spatial ontologies: **probabilistic** (or "fuzzy") and **binary** (or "true"). In the former, graph edges have associated weights, wherein higher weights represent that a given containment relation is more common or intuitive – e.g., one would expect that the weight of the edge from "toilet" to "bathroom" would be high, while "toilet" to "forest" would be much lower. In the latter, edges are accordingly unweighted and indicate that a lower-level concept can be found in a higher one (said relation is either present or not, hence *binary*). We discuss methods for creating both these ontology types, including a method for converting the former into the latter.

4.2.2 Constructing the Ontology Graph

For both probabilistic and binary ontologies, we first need to construct the spatial ontology graph. The method for doing so is a design choice that is up to the discretion of the user. For this work, we choose to use an off-the-shelf knowledge base to provide preliminary objects, locations, and spatial relations. After curating a list of objects that a pre-trained semantic segmentation network can detect, we query ConceptNet [87] to provide a list of

locations said objects can be found in by querying for the *atLocation* relation. We then query ConceptNet for the locations *of the initial list of locations* as well, thereby yielding more locations at higher levels of abstraction. From here, we now have a list of concepts that will act as the nodes in our ontology graph.

We manually divide these concepts into six groups, listed in ascending level of abstraction: (0) objects; (1) rooms; (2) buildings, transportation infrastructure (e.g., roads and bus stops), and small man-made/natural structures; (3) institutions or groups of buildings; (4) large-scale nature or settlements; and (5) world (consisting of a single world node). We assume that a node in a given level could be connected to/found in a node in *any* higher-abstraction group.[1] We thus procedurally generate directed edges from each node to *all nodes in all higher-level groups*. This naturally means some non-sensical containment relations will be generated, but said relations will ideally be assigned low weights (as described in the next section). These can either be kept (in the probabilistic case) or filtered out (in the binary case).

Note that this approach for constructing the ontology graph still requires significant hand-design. This is largely due to deficiencies with ConceptNet – when querying for locations, we note many typos, disjoint or one-off synonyms (e.g., "garden" and "flower garden" are disjoint concepts), and non-sensical/incorrect location concepts. Additionally, the relation edges in ConceptNet are sparse, with the attached weights simply being concept "relatedness." There is no clear canonical interpretation of these values in the context of scene understanding. Altogether, of the ~ 600 concepts yielded from calling *atLocation* in ConceptNet, we pruned ~ 400 for being nonsensical or irrelevant to spatial ontology construction. Then, of the 221 remaining concepts, we manually merged synonyms, yielding 128 unique concepts.

[1] It does not make sense to limit nodes to only connect to the next highest level of abstraction. For example, an umbrella (0: object) could be in a house (2: building) *or* on a beach (4: large-scale nature). We cannot have beach and house on the same level of abstraction, since a house can likewise be found on a beach. This necessitates our "skip connections."

4.2.3 Generating Ontology Weights

With the considered concepts and edges chosen, we now move on to an automated way of assigning weights to the containment relations. We decide to compute these scores via autoregressive language models, which learn to assign probabilities over text strings. Formally, let $\Lambda : W \to \mathbb{R}$ be the language model function mapping from arbitrary-length strings of tokens $W = \{W_1, W_2, ...W_N\}$ to the (log) probability of the string, as decomposed by the chain rule:

$$\Lambda(W) \approx \log p(W) = \log p(W_1) + \sum_{i=2}^{N} \log p(W_i \mid W_{1...i-1}) \tag{4.1}$$

This should assign higher scores to more semantically- and grammatically-likely sentences. We use the probability of a string $W_{A \to B}$ expressing "A is found in B." to be a proxy for the probability of being located in B given the presence of A. For each directed edge $(A \to B)$, we set the weight $w_{A \to B}$ as:

$$w_{A \to B} = \frac{\exp\{\Lambda(W_{A \to B})/k\}}{\sum_{B'} \exp\{\Lambda(W_{A \to B'})/k\}} \tag{4.2}$$

where the softmax is over all nodes B' in a higher level than A (i.e., all nodes that A has an outgoing edge to) and $k \in (0, \infty]$ is the Boltzmann temperature parameter. This score can be informally interpreted as $p(\text{in } B \mid A \text{ present})$ (shorthanded to $p(B \mid A)$), meaning the weights for all outgoing edges for any A sum to 1. Note that this is similar to the approach taken with LaMPP, as discussed in Chapter 3.

Such scores are computed for each edge in the ontology graph. This produces the full probabilistic ontology: a graph wherein each node represents a concept label in one of the six hierarchical layers (described in Section 4.2.2) and has outgoing edges to all higher-level nodes, each with an edge weight computed from language model evaluations (and are normalized per source node).

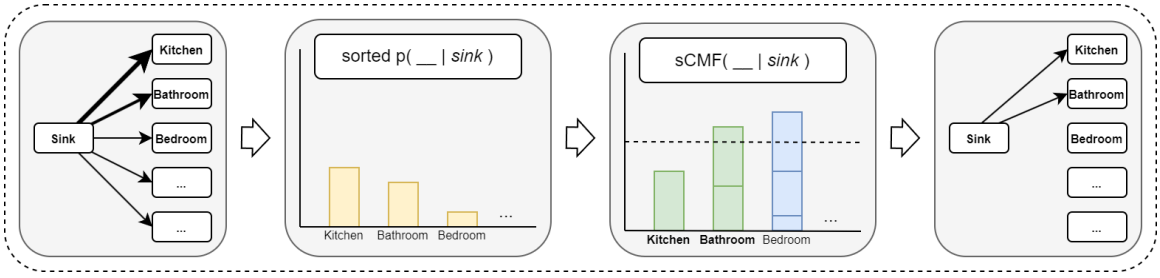

Figure 4-1: Example of how a subgraph of a probabilistic ontology can be binarized via sorted cumulative mass function thresholding. The weights (thickness) of outgoing edges of "sink" define a conditional distribution which can be turned into a sCMF function. After thresholding, only admitted relations' edges are kept in the binarized ontology.

4.2.4 Binarizing Ontology Edges via Thresholding

Once the probabilistic ontology has been created, it can be converted into a binary ontology. A naive method for doing this is by probability thresholding: include all edges with weight (probability) above some threshold. This method has two main problems. First, nodes in lower layers have more outgoing edges than ones in higher layers, meaning it is possible that the lower node's edges tend to have lower probabilities (on account of the distribution supporting a larger set). Thus, a probability considered high for edges from the lower layer can be considered low for edges originating in a higher one – a single threshold cannot accommodate this. This can easily be solved by assigning different thresholds for different layers, with lower layers having lower thresholds.

Second, this technique does not consider the relative magnitude of edge weights for a single node. For example, consider two nodes on the same level, each with 10 outgoing edges. If one node's outgoing edge weights are $[0.85, 0.12, 0.01, 0.01, ...]$ while the other's are all around 0.1, it intuitively seems like the first node should only have the highest-weight edge post-binarization (on account of it having much higher weight than all others) while the second should have many of its edges included.

This cannot be done with pure probability thresholding, so we consider another binarization technique that we call sorted cumulative mass function (sCMF) thresholding. For each node A, we sort its outgoing edges by descending weight $p(B \mid A)$. Then, we compute the cumulative mass function for these edges in said order. We binarize by iterating through the sorted edges and keeping all of the ones up to and including the edge whose

CMF exceeds some threshold.[2] This method still requires we choose different thresholds for different layers. However, by considering the *sorted* CMF, if a single edge weight dominates the rest, only it will be included (while flatter distributions over possible target nodes will include more of the corresponding edges). See Figure 4-1 for an example.

4.2.5 Direct Generation of Binary Ontology Edges

Alternatively, one can directly generate binary edges via a generative language model. For instance, one can ask a large generative language model trained for instruction-following via reinforcement learning for human feedback (RLHF) [69] to output locations where a given concept can be found.

4.3 Experiments and Evaluation

4.3.1 Querying for Language Model Scores

We experiment with different query prompts and strategies for generating ontology weights using GPT-J [95]. Once local evaluations resulted in sufficiently good results, we query for a final set of weights from the 175 billion parameter iteration of GPT-3 [13] to investigate if increased training and model size impact results. We find that the resulting probabilities $p(B \mid A)$ tend to have much lower entropy than other models (perhaps due to the significantly higher parameter counts enabling the model to learn much "spikier" distributions), so we set the Boltzmann temperature parameter to $k = 10$ in order to smooth them out.

We test five query structures, most of which use few-shot prompting [13]:

1. **Default**: Just the considered relation string, "*[concept 1]* is found in *[concept 2]*."

2. **Prompt 1**: Three fixed positive examples of containment relations followed by the considered relation.

[2] We include the one that exceeds the threshold so that all nodes have at least one outgoing edge. Otherwise, if the first edge's probability is very high, the only way to include it would be if the threshold were set even higher than that, which might mean that other nodes have *too many* edges included.

3. **Prompt 2**: Same as Prompt 1, but with a prefix describing the task: "Please tell me where common objects and locations can be found."

4. **Prompt 3**: Same as Prompt 2, but with 8 few-shot examples instead of 3.

5. **Prompt CN**: Same as Prompt 2, but few-shot examples are drawn from where ConceptNet says *[concept 1]* can be found.

4.3.2 Querying for Direct Generation of Binary Ontology Edges

We construct a binary ontology via direct-generation by querying ChatGPT [68] with prompts of the following form: "Consider the following locations: *[List of higher-level concepts]*. Of these locations, where can I find a *[consider concept]*? Please answer with a list of locations." This is done for each concept in the ontology, with each one yielding a structured list of locations a given concept can be found.

Note that this method requires some post-processing. While ChatGPT empirically almost always returns a list of concepts as requested, said lists often contain (i) extraneous prefixes, suffixes, and element-wise explanations and (ii) hallucinated location concepts (ones that are not in the specified list of considered higher-level concepts). These problems can generally be fixed programmatically (e.g., removing all text before and after the list, removing all text following list elements that follow colons or are in parentheses, and removing list elements that are not in the list of higher-level concepts). However, the hallucinations can often yield *incorrect* list elements as well. This is a known limitation of contemporary language models [37] – there is little way of dealing with this other than by treating such relations as noise.

4.3.3 Evaluating Human Judgement Alignment

Probabilistic Ontologies

We provide two metrics for measuring how well human judgements on concept relations are reflected by language model scores in probabilistic ontologies (relative to one another).

ConceptNet Precision at k: First, we consider the containment relations acquired from ConceptNet. While said relations are both too sparse for our purposes and only contain positive samples, if the language model scores are reasonable, they should rate the ConceptNet relations highly. We measure this by computing the average precision at k (p@k) of all ConceptNet containment relations between concepts in our ontology. For each such relation "A is located in B," we check if $w_{A\to B}$ is within the top k highest scores of A's outgoing edges, $\{w_{A\to B'}, \forall B'$ of higher level than $A\}$, reporting what portion of all such relations from ConceptNet fit this criteria.

However, we note that certain concepts have many *atLocation* relations according to ConceptNet – sometimes more than k, meaning it would be impossible for all such relations to the be in the top-k language model scores for said concept (so average p@k could never be 1). Moreover, many *atLocation* relations in ConceptNet are non-sensical or irrelevant, even when between concepts in our ontology. To fix these problems, we note that a concept's *atLocation* relations are ordered. The ones earlier in the list tend to be more intuitive or correct. Thus, we also measure average p@k for the top $N \leq k$ *atLocation* relations per object, thus only considering more sensible ConceptNet relations while making a perfect average p@k (or 1) achievable again.

We report the values in Figure 4-2. We note that all three fixed-prompt few-shot queries (Prompt 1, 2, and 3) all yield very similar results, with Prompt 2 being slightly higher than the rest. The adaptive ConceptNet prompts do worse than these three query types in all metrics. All query types yield universally better scores than the zero-shot default, indicating the importance of few-shot examples.

Coarse Human Judgement Evaluation: As ConceptNet relations are sparse, strictly positive, and often low-quality, we hand-annotate containment relations as being likely, plausible/sometimes-occurring, or unlikely, in accordance with human intuition on inferring high-level location information given the presence of low-level concepts. That is, the annotations reflect human estimations of the distribution $p(\text{in } B \mid A \text{ present})$ discussed in Section 4.2.3.[3]

[3]For example, we judge the statement "A toilet is found in a university" as "plausible/sometimes-occurring" instead of "likely" because the presence of a toilet only weakly implies one *might* be in a university (while strongly implying one is in a bathroom), *not* because only some universities have toilets (which we

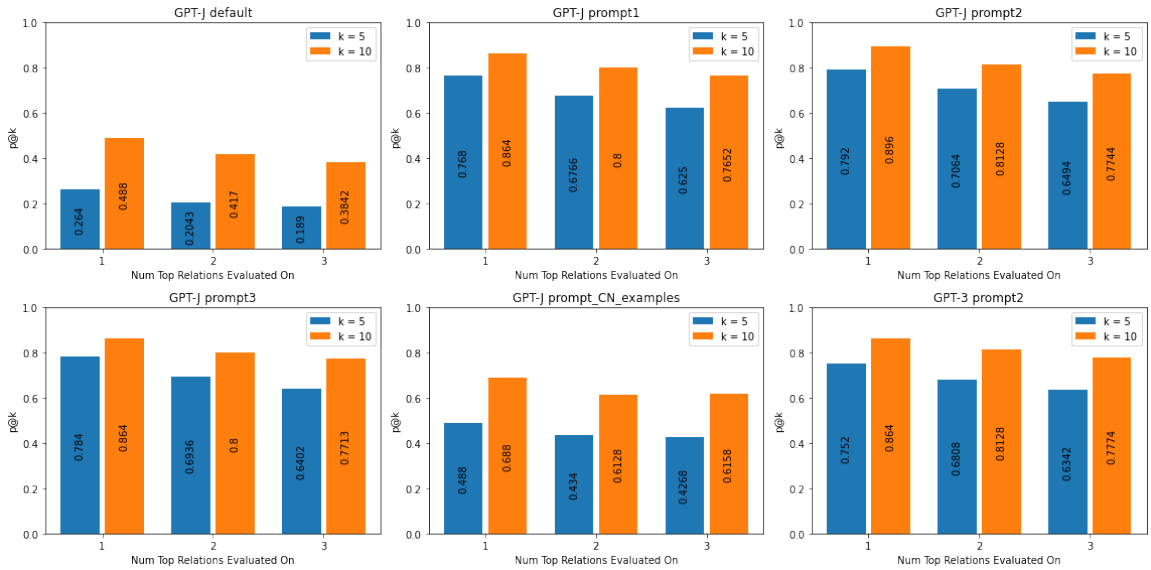

Figure 4-2: ConceptNet p@k ontology weights generated from GPT-J (with various queries) or GPT-3. Each subplot represents a different query type. x-axis represents N and the color of bar indicates $k = 5, 10$ for blue and orange respectively.

Due to annotator constraints, we annotate 15 containment relations per concept (or as many as possible, if the concept has < 15 possible locations). In particular, for each concept A, we look at the 5 top-, middle-, and bottom-scored relations $w_{A \to B}$ from GPT-J. This way, we get a diverse sampling of positive (or plausible) *and* negative relations based on language model judgements. Note that, while annotating, it is unknown which of these three groups a given relation is from; the annotator thus labels based off their own intuition, unbiased by what the language model might say. After removing relations with unclear label, the dataset contains 2444 containment relations and associated human judgements.

With this data, we divide the relations by label (likely/plausible/unlikely). For each group, we compute the *average percentile* of the score for each relation $A \to B$ relative to all possible $A \to B'$ (i.e., a relation $A \to B$ is in the 100th percentile if it is scored more highly than all other $A \to B'$ for given A). For a language model to match human judgements, we expect the "likely" group's relations to have a high average percentile, "plausible" to have semi-high/mid-range average percentile, and "unlikely" to have low average percentile.

very much so hope is not the case).

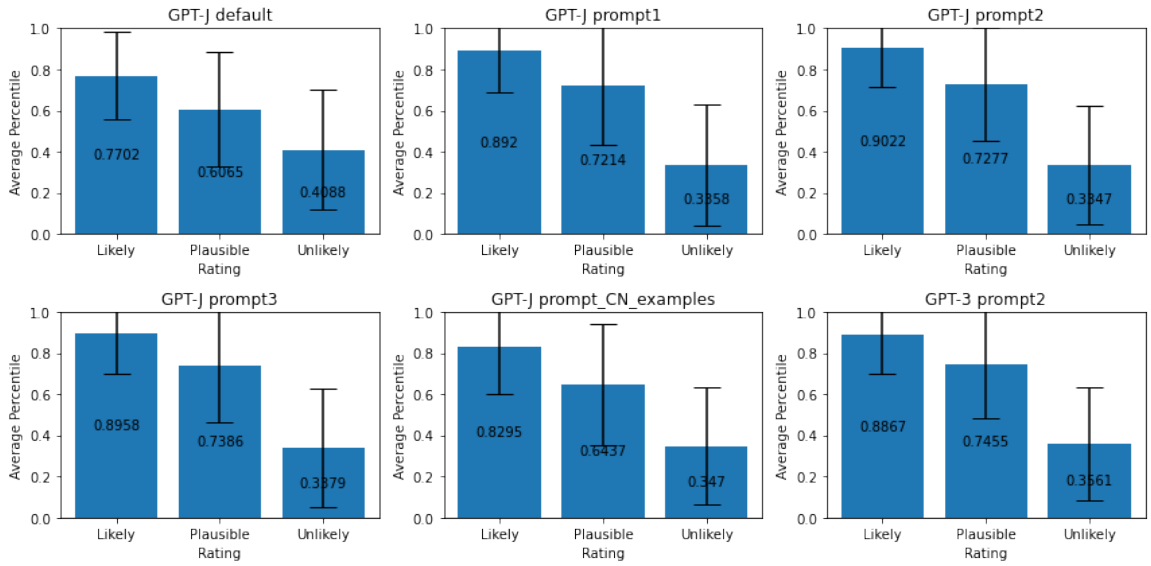

Figure 4-3: Average percentile of ontology scores of human relation judgements by rating for GPT-J (with various queries) and GPT-3. Error bars are one standard deviation.

We present these evaluations in Figure 4-3. As with the ConceptNet p@k metric, fixed few-shot prompts yield the best results, with all variations of said strategy being similar in performance. As expected, likely and plausible relations tend to have significantly higher average percentile than unlikely ones. Nevertheless, the unlikely percentile is still consistently above 30%, likely because there are more unlikely relations than not (so they must occupy a larger range of percentiles). This is the case for all three ratings, as indicated by the significant percentile variances.

GPT-3 Querying and Evaluation: Ultimately, while all Prompt 1, 2, and 3 yielded similar results and beat Prompt CN and Default, Prompt 2 consistently yielded slightly better values than the others (while not using as many tokens as Prompt 3). We thus query GPT-3 for ontology weights using this query type. The resulting probabilistic ontology has very similar (or even slightly worse) performance to its GPT-J equivalent's, as shown in the final subplot in Figures 4-2 and 4-3.

Binary Ontologies

Our coarse human evaluations also enable us to both tune and evaluate our binary ontologies. A highly-aligned binary ontology should have edges representing relations deemed

Model	Level	Best Threshold	Accuracy	True Positive	True Negative	False Positive	False Negative
GPT-J	**0**	0.3646	86.80%	79.37%	92.23%	7.77%	20.63%
	1	0.1325	85.56%	76.67%	89.76%	10.24%	23.33%
	2	0.4096	86.79%	80.06%	91.56%	8.44%	19.94%
	3	0.4375	78.51%	62.00%	90.14%	9.86%	38.00%
GPT-3	**0**	0.3905	86.27%	74.58%	94.82%	5.18%	25.42%
	1	0.3072	85.83%	76.67%	90.16%	9.74%	23.33%
	2	0.5705	86.54%	83.04%	89.03%	10.97%	16.96%
	3	0.5742	81.82%	68.00%	91.55%	8.45%	32.00%

Table 4.1: Best GPT-J and GPT-3 sCMF thresholds per-hierarchy level when optimizing for classification accuracy on the human judgements dataset. Layers contain: (0) objects; (1) rooms; (2) buildings, transportation infrastructure, and small structures; and (3) institutions or groups of buildings.

by humans as feasible (likely or plausible) while *not* containing edges for relations judged infeasible (unlikely). As such, the evaluative metric of interest is classification accuracy of these relation groupings by the presence of edges in the binary ontology.

Tuning Binarization Parameters: The human data aids in tuning binarization parameters. We do this for the sCMF method by finding what sCMF threshold value $\in [0, 1]$ (for each concept abstraction level) yields the best feasible/infeasible classification accuracy on the relations in the human judgement dataset. Note that this is simply one natural choice of optimization objective. Other objectives can also be used (for example, a weighted sum of true positive and true negative, wherein weights reflect whether accuracy on feasible or infeasible relations is more important to the user). For more details on this tuning method, see Appendix C.1.

We present the tuning results for GPT-J and GPT-3 in Table 4.1. We note two points of interest regarding the values. First, a general trend is that edges originating in higher levels of abstraction require higher sCMF threshold values. This agrees with the reasoning discussed in Section 4.2.4, wherein low-level concepts have more outgoing edges (and thus on average have lower weight edges), meaning a lower sCMF threshold can be used to admit a comprehensive valid binarized edge set.

The only exception to this trend is for edges originating from the object layer. This leads to the second point: the object-level optimal computed threshold does not admit empirically good relations – in practice, it admits *too many* relations. We suspect this is

Binary Relations	Accuracy	True Positive	True Negative	False Positive	False Negative
ConceptNet	69.89%	25.96%	99.66%	**0.34%**	74.04%
GPT-J w/ sCMF	**83.04%**	68.76%	**92.71%**	7.29%	31.24%
GPT-3 w/ sCMF	82.63%	70.18%	91.07%	8.93%	29.82%
ChatGPT w/ direct-generation	78.86%	**71.70%**	83.71%	16.29%	**28.30%**

Table 4.2: Classification accuracy and confusion matrix metrics of various binary ontology production methods when evaluated on human relation judgement data. Bolded values are best score per metric (higher is better for accuracy and true classifications, lower for false classifications).

because all concepts only have ~ 15 ground truth evaluations associated with them. Thus, it does not adequately reflect all possible relations represented by object nodes' outgoing edges (of which there are over 100 per node). We therefore expect significant error between the true optimal threshold and the one computed here. The only way to lower this error is to acquire more human judgements for said relations. As higher-level concepts have fewer outgoing edges, this is less of a problem.

Thus, we binarize the probabilistic ontologies generated via GPT-J/3 by using the optimal-accuracy thresholds shown in Table 4.1 for all levels *except* the edges originating from object nodes. For that, we set the sCMF threshold to $0.1, 0.2$ for GPT-J/3 respectively instead, which empirically seems to admit most feasible relations while rejecting infeasible ones. Further, note that edges starting in level 4 are always admitted, as level 4 nodes are always only connected to the single level 5 world node.

Evaluating Binary Ontologies: Once binary ontologies have been created (via any method), they can be *evaluated* on the human judgement dataset too. In this case, we measure both overall classification accuracy (wherein ground truth feasible relations are correctly classified if a corresponding edge is in the binary ontology whereas infeasible ones are not) and standard confusion matrix metrics. We do this for our GPT-J and GPT-3 sCMF thresholding and ChatGPT direct-generation binary ontologies. We also evaluate ConceptNet's *atLocation* relations against the human dataset.

These results are shown in Table 4.2. They indicate that ConceptNet yields worse accuracy than all other methods. This is because of its sparsity – after pruning and merges,

it has 678 feasible containment relations, compared to GPT-3's 1878 or ChatGPT's 4634. Thus, it classifies a vast majority of ground truth relations as infeasible/negative. This is shown in the significant false negative rate (74.04%).[4] Of the relations it *does* classify as feasible, very few of them are incorrect (0.34% false positive rate), so ConceptNet's positive classifications tend to be *valid and aligned with human judgement*, but due to sparsity, it only achieves 25.96% true positive rate. Overall, this sparsity (in conjunction with the significant amount of manual refinement needed to make ConceptNet usable) affirms the fact that ConceptNet is impractical for use as a spatial ontology out of the box.

As for the language model approaches, it seems like the sCMF ontologies from GPT-J and GPT-3 are extremely similar, with the former yielding slightly better results. Further parameter tuning may change this, but with just the human data tuning described prior, GPT-3's magnitudes-larger model size does not seem to make a significant performance impact. Likewise, when comparing between sCMF and direct-generation, both ChatGPT and GPT-3/J yield similar performances. The primary difference seems to be that, unlike ConceptNet, ChatGPT's ontology has *many* edges (i.e., many positive/feasible classifications). This tendency to overpredict positive relations leads to the best true positive and false negative performance, but likewise causes degraded true negative and false positive rates. In turn, because the dataset is mostly negative relations, GPT-3/J's higher true negative rate yields overall higher accuracy.

4.4 Discussion

4.4.1 Key Takeaways

We cover some key qualitative takeaways, both from our evaluation data and from experimental experience.

First, ConceptNet is far too sparse to cover the broad spectrum of human spatial ontology judgements, so other tools should be used to provide denser judgements. Based on the

[4]While worse than other methods', the ConceptNet overall accuracy is still relatively high (70%) because most ground truth datapoints *are* negative relations, so the negative over-prediction caused by sparsity actually is often correct (as shown by the 99.66% true negative rate). As the dataset consists of $\sim$ 60% infeasible relations, a classifer that exclusively outputs "infeasible" would get 60% accuracy.

above results, all our language model ontology-construction methods seem fairly aligned with human judgements, and are thus a suitable candidate for supplementing knowledge base data. In terms of binary ontologies, direct-generation via ChatGPT and sorted CMF thresholding of relationship probabilities with GPT-3 yield similar performance on human judgement evaluation metrics.

The decision of which method to use therefore must consider the following practical trade-offs. Direct-generation just involves passing queries to a language model and using its generated text to construct the ontology. Thus, it is *much* easier to use than sCMF thresholding, which requires much more parameter tuning. However, direct-generation also requires some data post-processing to turn semi-unstructured strings into symbolic ontology representations. Said post-processing should account for general problems with modern generative autoregressive models, such as hallucinations, occasional misinterpretation of/sensitivity toward prompts,[5] and incorrect/useless outputs.

On the other hand, sCMF thresholding's parameters can also be a strength: one can use supplementary data for tuning, thereby leading to more-alignment and better performance. Additionally, its paradigm of evaluating small "relation fact" strings allows for reusable and expandable ontology data; when new concepts are added, one can simply evaluate new possible relations via the language model while keeping existing evaluations. Direct-generation instead requires re-querying the language model to regenerate the whole ontology with the updated prompt, which is much more (financial and compute) resource-intensive.

4.4.2 Possible Evaluation Concerns

One concern with the binary ontology evaluation is that our methods are both tuned and tested on the same human judgements dataset. We claim this is actually not a problem. The main use of held-out training splits for learned models is to show that said model

[5] E.g., it seems like ChatGPT tended to predict concepts at the start or end of the list of possible concepts more often than ones in the middle. This tendency aligns with human psychology as well [64]. For instance, most of the rooms were close to the front of the list *except for bathroom*, which was in the middle. Said other rooms are often reliably and correctly predicted, but bathroom was more inconsistent, even for objects that nearly exclusively appear there, like showers or bathtubs.

generalizes to new (usually in-distribution) data, rather than over-fitting to and memorizing the training set. The key difference of our problem is that human judgement datapoints are *not* sampled from a distribution – they instead reflect a subset of (assumed-ground truth) relations present in some underlying "true" spatial ontology.

For example, if the ground truth says chairs can feasibly be found in classrooms, that relation is present in the "true" ontology; one would not be able to sample another true ontology relation that says that chairs *aren't* feasibly found in classrooms. Critically, this is different from saying that grounded scenes must always obey these ontology relations – just because some particular sampled classroom does not have chairs does not change the ontological fact that, in general, *chairs are feasibly found in classrooms*.

Memorization of the ground truth relations is thus not only harmless, but *actively beneficial*. If any data is held out to show that the created ontology can generalize to it, said ontology can *always be improved* by incorporating that split into the tuning set and having the ontology fit to it.

4.5 Future Works

We now discuss further ways of extending the automatic spatial ontology construction process and actual applications of the data structure.

4.5.1 Further Automation of Ontology Construction

There are several points in the above methods that require significant human hand-design. Namely, at the start, we (i) pick out concepts to include in the ontology and (ii) arrange them into a hierarchy. We now outline a process for automating the second point. Note that the following method can be seen as an extension of the approach in [17].

Two simple starting points are (i) using a language model to evaluate *all possible concept pairs'* containment relations $p(B \mid A)$ and keeping some subset of relations that form the node hierarchy and (ii) asking a generative language model to list possible locations for a given concept out of the list of all other concepts. We call these two methods LaMPP

evaluations (as we treat language probabilities as proxies for relation priors, as we do in Chapter 3) and direct-generation respectively.

Of course, neither of these resulting graphs are necessarily hierarchical, which we formally define as a multipartite directed graph wherein (i) the node partitions have some order and (ii) edges from nodes in one partition can only point to nodes in later partitions. This definition of hierarchy is equivalent to the ontology being a directed acyclic graph (DAG). For a simple proof, see Appendix C.2. Thus, we require a way to make an automatically-generated ontology graph into a DAG (which then gets turned into a hierarchy), as there is no guarantee that either aforementioned method's graph is a DAG (with the LaMPP evaluation approach yielding a fully-connected graph).

To assist with the conversion, assume all edges in the initially-generated graph have associated probability weights. Direct-generation does not produce these weights, so simply apply LaMPP evaluation to all the relations the generative model produced. Now, we note that, since non-sensical relations tend to have lower language probabilities, the resulting DAG should get rid of low-scoring edges; conversely, it should keep the high-scoring edges.

This is therefore a suitable example of the *maximum acyclic subgraph* problem: given a directed graph with edge weights, produce a subgraph that is acyclic and of maximum possible total weight. This problem is NP-hard [41], but approximations exist [8, 32]. One simple $\frac{1}{2}$-approximation [32] with runtime $O(|E|)$ is presented with proofs of correctness and complexity in Appendix C.3.

This algorithm can further be improved by running it multiple times with different random permutations of the vertices $\in V$, then taking the produced subgraph with the highest total weight. If run k times, the runtime is $O(k|E|)$. Better approximations can be found by choosing the permutation carefully [32], but we consider this simple case for now. Moreover, any sorts of sparsification of the initial graph aids in the approximation – a simple source would be to remove all edges $u \to v$ with lower weight than $v \to u$, but if any other edges are known to not be possible or reasonable, they can be removed too.

We thus have a method for constructing a hierarchy out of (possibly cyclic) weighted ontologies. Given such a graph (e.g., produced via LaMPP evaluation or direct-generation),

run the above randomized algorithm to produce some $\frac{1}{2}$-approximate DAG (V, E') in $O(k|E|)$ time. Then, produce a corresponding topological ordering with an efficient topological sort, e.g., with time complexity $O(|V|+|E|)$ [21]. Lastly, use the topological ordering to compute the longest directed path (in terms of edge counts, not weights) from some node with no incoming edges to each node in $O(|V|+|E|)$ time. This assigns an integer longest path distance to each concept node, which can then be used as its hierarchy partition number. We note that, in practice, this should be done after some additional sparsification – otherwise, the longest paths from source nodes may be too long.

See Appendix C.4 for the full algorithm and associated proofs. With this hierarchy created, one can either treat it as a probabilistic ontology or binarize it (e.g., via sCMF thresholding or, if initially created via direct-generation, by simply removing all weights).

4.5.2 Use of Ontology in Spatial Perception

Once the spatial ontology has been created, it can be used for downstream learning tasks. In particular, we consider the use of the Logic Tensor Network (LTN) [6] framework for training graph neural networks to make predictions in accordance to ontology relations. The LTN encodes logic rules that the classifier should obey. Of particular relevance is the *inclusion* predicate, which enforces that a low-level node contained in a higher-level one should both have labels that represent a feasible relation (i.e., the spatial ontology says the low-level label can feasibly be found in the high-level location).

At training time, the neural network tries to output predictions that satisfy the collection of predicates as well as possible. In effect, this is a *loosening* of standard neural maximum likelihood class prediction: instead of just learning to output the exact labels seen at train time (which might be noisy anyway), the class predictions should simply be feasible/consistent with the spatial ontology's common sense.

Chapter 5

Future Works and Conclusions

5.1 Current Work Extensions

In this thesis, we considered how language models can provide abstract common sense for robot navigation and scene understanding tasks. In particular, we introduced (i) a variety of language-leveraging methods for room classification, including ones that leverage vision, (ii) a framework for using language model evaluations as priors in probabilistic graphical models, with particular applications to robot navigation, and (iii) a method for automatically extracting and codifying language models' common sense on object and location relations in a spatial ontology.

Each of these works still have many avenues to explore:

1. **Scene understanding:** Due to compute limitations, we were heavily restricted in terms of both the types of language models we could load and the types of approaches we could evaluate. Given a higher compute budget, we can try some of the experiments detailed in the Chapter 2 Limitations section: namely, more advanced language model techniques, like chain-of-thought prompting (asking the language model to describe a room before making a final classification) or Socratic models (having a VLM describe a room, then using a separate language model take in those descriptions and reason about them to classify the room). Even just experimenting with larger models could yield improved results. We thus would like to see how

improved compute could change our findings, even if our current results are quite promising.

2. **Navigation:** LaMPP adopts a simplified version of the ObjectNav challenge. It would be interesting to see how we could apply it to the original problem, wherein the map and room labels are unknown. In this case, it would have to perform internal mapping in conjunction with using the language model priors, possibly extending the probabilistic graphical model we currently adopted to account for unknown rooms. Additionally, as one of our LaMPP experiments evaluated how it could be used for semantic segmentation, another possible extension is to combine the navigation and segmentation LaMPP frameworks, e.g., to correct for mis-identifications of objects of interest.

3. **Ontology:** We are currently in the process of extending the ontology work to see how the data structure can be used for actual downstream learning, as mentioned in Section 4.5. The further automation methods listed there are also possible future research paths.

5.2 What Comes Next?

There are many other use cases for language models' common sense in robotics, beyond just navigation and scene understanding. As an example, we consider the problem of giving robots natural language feedback or corrections.

Currently, the primary approaches are to either train an end-to-end generative transformer to correct robot trajectories conditioned on language [14] or to learn mappings from language to shaped rewards [82], which can then yield policies via model-predictive control [65] or reinforcement learning.

The former approach seems much less scalable and reliable – one has to just trust the outputted trajectory is sensible and in-distribution. The latter seems more promising, though a standard supervised language-to-reward mapping may still be inefficient. Rather, by using language models' common sense, there may be a clever way of inferring the prag-

matics behind a piece of feedback or correction – that is, the underlying reason (or featural dependence) for why said feedback is received.

Indeed, linguistic feedback is generally viewed as providing partial policies or reward functions to a decision-maker; a teacher can tell the agent what to do in certain situations or the underlying value behind a state, with the latter being generally more useful in longer-horizon/lower-supervision tasks [88].

However, this distinction is not solid; even partial policies can yield underlying reward information [53]. In doing so, we would intuitively expect to improve sample efficiency of robot feedback: if a robot is told to avoid carrying a glass of water over a laptop, it might understand that this is because the laptop is an expensive electronic device, and thus generalize its avoidant trajectory behavior to other objects that should be treated in the same way (e.g., phones or tablet computers).

Such inferences could rely on language models' common sense. They likely capture information on the fact that laptops and phones are expensive and susceptible to water damage, so the language models can be a source of rich, non-obvious features for scene elements. Naturally, numerical judgement alignment tests (akin to the ones in Chapter 4) should be employed to determine how well such features and understanding match that of humans.

Going even beyond that, there has been recent success in bootstrapping desirable language model assistant behaviors from a small sample of examples and guiding principles [89]. This strategy could also be applied to pragmatic robot corrections, e.g., pick a few sample corrections (and maybe associated underlying reasons behind the feedback) and have the LM generate more, based on either the same pragmatic reasons or other intuitively viable ones. Such corrections can then either be applied for reward shaping (as with [82, 65]) or even to train a policy network with the guiding principles engrained within, as is done with the language model in [89].

5.3 Conclusion

In this work, we present an overview of methods for leveraging language models in robot navigation tasks and for scene understanding on both small and large scales. In doing so, we thoroughly examined the strengths and deficiencies of modern language models – they are a useful and versatile source of common-sense prior knowledge which is both actionable and aligned with human judgements, but also suffer from standard language model failure cases, like hallucinations, uncalibratedness, and prompt sensitivity. Moreover, as this work shows, using language models for robotics introduces its own host of challenges: how does one project observation or state information into string space while only keeping relevant information, if one should do so at all? How do we ground language in real scene elements, or, perhaps even harder, actions of robots and humans?

We do not have answers to these questions, and, quite frankly, it would be boring if we did. However, overall, a theme of both the above discussion and the research presented in this thesis is that current language models seem to work well as *intuition* instead of pure, logical/formal reasoners. As such, we argue they should be used in conjunction with symbolic or formal systems as a sort of common-sense guidance. We do this in all three projects to some extent:

1. **Scene understanding:** The LM takes in symbolic object labels and spatial geometry data from a spatial perception system (possibly in conjunction with images) to exceed the performance of pure-vision or GNNs operating on just the symbols alone.

2. **Navigation:** LaMPP fills in probabilistic graphical model parameters or relations with LM scores.

3. **Ontology:** The LM fills in relations between symbols in the ontology graph, whose relations are then engrained in LTN training.

Such an admittedly broad framework is attractive due to how language models and formal symbolic systems complement each other: symbolic systems operate on rigid rules, enabling so-called "system 2" cognition, but lack scalability and generalization. Language models (and many other neural methods) encode and operate on "fuzzy" knowledge, which

can operate on a wide array of concepts, but does not have built-in formal guarantees as to how these concepts are connected; they are akin to "system 1" thinking [40]. Likewise, there are situations wherein one wants a robot to act rigidly and precisely. On the other hand, there are domains wherein it is difficult to write down formal/logical rules for robot behavior. Through this lens, successful integration of language models into robotics can be summed up as finding the right balance of these qualities.

Regardless, we have demonstrated some early successes in language in robot scene understanding and navigation. It is a good and exciting time to see what comes next in this disciplinary intersection.

Appendix A

Robot Scene Understanding Implementation Details

A.1 Converting Matterport3D to Scene Graphs

To convert semantic meshes from Matterport3D into scene graphs, we create a node for each region and object [15]. Then, we connect all object nodes assigned to a region to that region's room node. We also filter out some regions. While Matterport3D contains outdoor regions as well ("yard," "balcony," and "porch"), we do not perform inference over them, since they are not true rooms and thus would require an alternate query string structure. In addition to outdoor regions, we also remove all rooms with no objects within or with the label "none."

Each object is assigned labels from several label spaces. We consider the original labels used by Matterport3D (mpcat40) and the labels used by NYU (nyuClass) [84]. For both, we filter out nodes belonging to the mpcat40 categories "ceiling," "wall," "floor," "miscellaneous," "object," and any other unlabeled nodes. We remove these categories because they are either not objects within the room or they are ambiguous to the point of being semantically uninformative. However, for nyuClass, we do *not* reject objects classified by mpcat40 as "object," since nyuClass has many more fine-grained and semantically-rich categories which all are mapped to this category. After pre-processing the label spaces in this way, mpcat40 has 35 object labels and nyuClass has 201. Both datasets share a room label

space with 23 labels. See Table A.1 for a breakdown of room label frequencies.

Room Label	*Bar*	*Bathroom*	*Bedroom*	*Classroom*	*Closet*	*Conference Auditorium*
Occurrences	3	365	251	2	99	16
Percentage	0.16%	19.43%	13.37%	0.11%	5.27%	0.85%
Room Label	*Dining Room*	*Family Room*	*Game Room*	*Garage*	*Gym*	*Hallway*
Occurrences	74	61	17	14	16	326
Percentage	3.94%	3.25%	0.91%	0.75%	0.85%	17.36%
Room Label	*Kitchen*	*Laundry Room*	*Library*	*Living Room*	*Lobby*	*Lounge*
Occurrences	78	35	1	71	62	64
Percentage	4.15%	1.86%	0.05%	3.78%	3.30%	3.41%
Room Label	*Office*	*Spa*	*Staircase*	*Television Room*	*Utility Room*	***Total***
Occurrences	98	44	152	13	16	**1878**
Percentage	5.22%	2.34%	8.09%	0.69%	0.85%	**100%**

Table A.1: Room label frequencies in pre-processed Matterport3D dataset.

We perform a few additional dataset pre-processing steps to produce the final scene graph dataset with 1878 rooms. First, since some objects are assigned an incorrect region (e.g., toilets are assigned to living rooms, despite (i) that being non-sensible and (ii) the toilet not being within the bounding box of the living room), we check to see if each object is within the bounding box of its assigned region. If not, then it is re-assigned to whichever region's bounding box contains it, and the corresponding scene-graph connection is also made. Second, nyuClass has some misspelled labels (e.g., "refridgerator" instead of "refrigerator"), so we correct all of those too. Lastly, sometimes, a single nyuClass label may be erroneously assigned to multiple mpcat40 labels. This is most problematic when one of the mpcat40 labels is rejected and the other is not. To address this, we use the first mpcat40 label for each nyuClass label that is *not* rejected (e.g., nyuClass label "stairs" is mapped to mpcat40 "miscellaneous," which is rejected, and "stairs," which is not, so we keep the latter). However, this means some labels which *should* be rejected are not rejected, so we also manually filter out all nyuClass object labels that are the *same* as those of rejected

mpcat40 labels: “ceiling,” “floor,” and “wall”.

A.2 Embedding-based Bootstrapping Method

To generate training data for the embedding-based method, we take the n most informative objects in each room and find all k-object permutations, producing P_k^n query datapoints per room of the form in Eq. 2.8, all of which correspond to the room's label. We do this for $(k,n) \in \{(1,2),(2,3),(3,4)\}$. Models trained on this data will thus be *invariant to object order and number* in the query, and can also handle less informative object labels.

A.3 Structured-language Query Details

To fit hardware memory constraints, we omit rooms with over 100 objects and round all values to three decimal places. Note that a given room might have different objects depending on which object label space is considered, as nyuClass includes labels that would be just classified as “object” by mpcat40 (which we reject).

A.4 Vision and Language Dataset Generation

To construct a dataset of room images for our vision and language methods, we use Matterport3D's Pathfinder functionality to generate a top-down occupancy grid map of freespace for a considered height (which we set to the height of a considered room). A single map pixel's length is set to 0.1 meters. We find all pixels that fall within a given room's bounding box, masking out the rest. To avoid sampling points that are too close to walls and objects, we dilate occupied cells with a 3×3 convolutional filter. Then, we uniformly sample (without replacement) 100 unoccupied cell positions (or as many as possible, if the room has fewer than 100 open cells). Each one is assigned a yaw uniformly at random. Finally, we set the camera to each of those poses, saving the viewed images and corresponding room label.

A.5 Training Details

For the *GraphSage baseline*, we train for 500 epochs with a learning rate of $5e-3$, weight decay of $1e-4$, hidden state dimension of 16, dropout of 0.2, and 2 iterations of message-passing.

For the *embedding-based approach*, we train each network for 200 epochs with a batch size of 512 using cross entropy loss via the Adam optimizer with a learning rate of $1e-4$, $\beta_1, \beta_2 = 0.9, 0.999$, weight decay of $1e-3$, and a StepLR scheduler with step size of 10 and $\gamma = 0.5$ [43].

For the *structured-language approach*, we train the T5 LM for 5 epochs with a batch size of 2 (due to memory constraints) and the AdamW optimizer with learning rate of $1e-4$, smoothing term $\varepsilon = 1e-8$, and no weight decay [55].

For the *ResNet baseline*, we train for 10 epochs using the SGD optimizer with a learning rate of $1e-3$, a batch size of 128, and momentum of 0.9. We load the checkpoint weights with the highest validation image-wise accuracy.

For the *fine-tuned VLM approach*, we train a BLIP-2 model for 5 epochs with the AdamW optimizer with a learning rate of $5e-6$, batch size of 8 images (due to memory constraints), weight decay of 0.01, and smoothing term of $1e-8$.

In trials training with the vision and language dataset, we pick 16 random images per room (as many as possible if there are fewer than 16). Additionally, for both validation and testing, we take every fourth image. Both these choices are to reduce the training and evaluation time – more images can be used for a more complete evaluation.

A.6 Building Embedding-based Data Generation Details

To generate data for inferring building labels via the feed-forward approach, for each building in the train set, we sample (without replacement) and shuffle $k = 4$ rooms out of the present ones, subsequently putting them into query strings, which we then embed via language model. The probability of choosing each room is proportional to the number of times that room appears in the building. We do this 1000 times total for each room, mean-

ing the training dataset label distribution has the same ratios as the training dataset used by GraphSage.

Appendix B

LaMPP Baseline Details

B.1 Model Chaining Navigation Baseline

We develop a model chaining baseline to compare against LaMPP, akin to [1]. Given a list of room labels and counts for the current environment, we query the language model to autoregressively fill in the string:

$$\begin{aligned} W = &\text{“The house has [}\textit{list of rooms with counts}\text{].} \\ &\text{You want to find [}g\text{]. First, go to each [}r_0\text{].} \\ &\text{If not found, go to each [}r_1\text{].} \\ &\text{If not found, ...”} \end{aligned} \tag{B.1}$$

where $r_{0,1,...}$ are room labels in the list of all present rooms (with no repetitions). The agent visits all r_0 in order of proximity. If goal object g was not found, then it visits all r_1 in the same way. This repeats until all rooms have been visited or the agent thinks it detected the desired object..

Appendix C

Ontology Construction Proofs

C.1 Sorted CMF Parameter Tuning with Human Data

We discuss specifics of how we tuned GPT-3's sCMF binary ontology parameters. For each layer of our hierarchy, we compute sCMF values for all outgoing edges of nodes in that level. We then sort all such edge relations (i.e., of all nodes in the layer) by ascending sCMF. Edges with smaller sCMF are ones that have relatively higher probability (of the outgoing edges for a particular node). Thus, they (i) should be kept post-binarization and (ii) *can be kept* with lower sCMF thresholds.

We then iterate through this sorted list of sCMF values for candidate thresholds. We compute what percent of the relations originating at the current level in the ground truth dataset would be correctly classified by a given threshold, i.e., feasible relations should have sCMF lower than the candidate threshold while infeasible ones should be higher. This allows us to find the candidate threshold that maximizes accuracy for each level. We then use this information to inform our layer-wise threshold choices.

C.2 Proof of Equivalence of Hierarchical Graphs and Directed Acyclic Graphs

We prove that a hierarchy is equivalent to a DAG with the following argument.

A hierarchy can trivially be converted into a topological ordering by iterating through the partitions in order and numbering the nodes in each partition in any desired order. When considering the subsequent partition, begin the numbering where it ended for the previous partition (e.g., if some partition's last node was numbered with n, the next partition's first node should be numbered $n+1$). Doing this for all nodes in all partitions yields a topological ordering: by definition of hierarchy, nodes can only have edges pointing to nodes in later partitions. Since nodes in earlier partitions were numbered before ones in later partitions, this means that all edges are from nodes with lower number to higher number, thus showing the ordering is topological.

Now to show the reverse. A topological ordering can easily be converted into a hierarchy by setting the number of partitions equal to the number of nodes and putting each node in the partition of number equal to its place in the topological ordering. This trivially satisfies the definition of a hierarchy – each partition only consists of one node (so the graph is multipartite) and only connects to nodes in later partitions (as their partition number is their topological ordering number, i.e., larger than the considered nodes).

This shows equivalence of hierarchical graphs and ones with topological orderings. We then know that graphs with topological orderings are equivalent to DAGs – topological orderings trivially disallow cycles (as it would involve an edge going "backwards" in the ordering) and topological orderings for a DAG can be produced via any topological sorting algorithm [21].

Thus, by the transitive property, since hierarchical graphs are equivalent to topologically ordered graphs and topologically ordered graphs are equivalent to DAGs, hierarchical graphs are equivalent to DAGs.

C.3 Analysis of the Approximate Maximum Acyclic Subgraph Algorithm

The simple $\frac{1}{2}$-approximation for maximum acyclic subgraph is:

Given $G = \{V = \{v_{1,\ldots,N}\}, E\}$

$E' \leftarrow \{\}$

for $i = 1, 2, ..., N-1$ **do**

  $V' \leftarrow \{v_{i+1,...,N}\}$

  if $w(\{v_i \rightarrow V'\}) > w(\{v_i \leftarrow V'\})$ **then**

    $E' \leftarrow E' \cup \{v_i \rightarrow V'\}$

  else

    $E' \leftarrow E' \cup \{v_i \leftarrow V'\}$

return (V, E')

where E contains weighted directed edges, $\{a \rightarrow B\}$ is the set of edges from a to nodes in set B (vice-versa for $\{a \leftarrow B\}$) and $w(\cdot)$ is the total weight of a set of edges.

This algorithm always returns an acyclic subgraph. This can be shown by contradiction. Suppose the returned graph was cyclic. Now consider the nodes in the cycle. This particular subgraph was produced from some fixed ordering/permutation of the nodes. Thus, there must be some node in the cycle such that all other nodes in the cycle come after it in the ordering. As the node is part of a cycle, it must therefore have both an incoming and outgoing edge, both connected to nodes that are *later* in the permutation. However, this cannot be the case for any subgraph returned by our algorithm, as in any given step, all edges added to E' are exclusively incoming or outgoing for the considered node and the ones after it in the ordering. The included edges from any node and the ones after it in the ordering must all must be incoming or outgoing – no mix of the two. This results in a contradiction, so (V, E') returned by the algorithm must be acyclic.

This algorithm is also a $\frac{1}{2}$-approximation, as all graphs (V, E') it yields must satisfy $w(E') \geq \frac{1}{2} w(E) \geq \frac{1}{2} w(E^*)$, where E^* are the edges of the optimal subgraph. This is because E' is comprised of the union of $\{v_i \rightarrow \{v_{i+1,...,N}\}\}$ and $\{v_i \leftarrow \{v_{i+1,...,N}\}\}$ (whichever has higher weight for each value of $i \in 1, ..., N-1$). The union of these $2(N-1)$ non-overlapping sets is E, and since E' contains sets who each have higher weight than unique members of the set of rejected edge sets, $w(E')$ must at least half $w(E)$, as desired.

Finally, this algorithm considers each edge's weight once (i.e., all the calls of $w(\cdot)$ are, again, for non-overlapping sets whose unions are E), so assuming $O(1)$ time complexity of accessing each incoming/outgoing edge for a particular vertex, it has overall runtime

$O(|E|)$.

We do not attempt to show any connectedness guarantees, but note that if the resulting subgraph is not weakly-connected, it can be made so by introducing an auxiliary node (equivalent to the "world" node) and adding edges from all nodes with no outgoing edges to it[1].

C.4 Longest Path Hierarchy Partition Algorithm

The longest path problem is known to be NP-hard for general graphs [41], but has a well-known linear-time dynamic programming solution for DAGs [47]. For such graphs, the longest path must always start at a source node (one with no incoming edges), as if it does not, then the path start node has at least one incoming edge, so the path can be extended. Thus, this problem is equivalent to finding the longest distance of all nodes in a DAG to (any) source node. Note that we consider longest path by edge count, which is a special case of the longest path by edge weight, but with all weights are set to 1. The algorithm is as follows: First, topologically sort the DAG G, if it has not been already. Call this order top_order(G). Then, execute the following:

for $v \in \text{top\_sort}(G)$ **do**
  if $v.\text{incoming} = \emptyset$ **then**
    $v.\text{dist} \leftarrow 0$
  else
    $v.\text{dist} \leftarrow 1 + \max_u\{u.\text{dist} \mid \forall(u,v) \in v.\text{incoming}\}$

Thus, if v has no incoming edges (i.e., it *is* a source node), then it has 0 assigned as its distance. However, if it has incoming edges, then its maximum distance to a source node must be the maximum over all its incoming edges' origin nodes' distances plus an additional edge. This maximum can be efficiently computed because all of the nodes with edges pointing to the considered v must have already had their distances computed, on account

[1] Although less easy to implement, one can also add rejected edges from the original graph to connect disconnected subgraphs until they are all weakly connected. This improves the total weight of the resulting graph while not introducing any cycles (e.g., if only a single edge is introduced to connect each disconnected subgraph to the "main" subgraph).

of going through the graph in topological ordering. The distance of subsequent nodes likewise cannot affect the current v's maximum distance, as they cannot have edges pointing to v (or any path to nodes that do have such edges), once again due to topological sorting. Thus, the assigned distance to each node is its true longest path to a source node, proving the algorithm's correctness. In terms of runtime, the algorithm iterates through each node once. For each one, it takes a maximum over parent nodes' distance scores, of which there are $|v.\text{incoming}|$. Thus, the maximums look at each edge once, yielding a runtime of $O(|V|+|E|)$. The topological sort has the same complexity, so the overall runtime does not change from it.

These distances are easily proved to be a valid hierarchy partitioning: nodes of the same distance cannot have edges between them (as the distance scores of a pair of nodes connected via an edge must differ by at least 1). Nodes' outgoing edges also definitely point to nodes in higher partitions, as an edge (u,v) necessarily means v's distance is greater than u's.

Finally, this strategy *minimizes the number of partitions*. To prove this, consider that this strategy produces k partitions, where k is the number of nodes in the longest directed path in G. If there were fewer than k partitions, by the pigeonhole principle, at least two of the nodes in that directed path must be in the same partition. As there is a path starting from one of those nodes to the other, this violates the fact that this is a hierarchical partition, as the nodes in later partitions would connect back to nodes in an earlier one.

Bibliography

[1] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, A. Herzog, D. Ho, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, E. Jang, R. J. Ruano, K. Jeffrey, S. Jesmonth, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, K.-H. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Rettinghouse, D. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, M. Yan, and A. Zeng. Do as i can and not as i say: Grounding language in robotic affordances. In *arXiv preprint arXiv:2204.01691*, 2022.

[2] J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds, R. Ring, E. Rutherford, S. Cabi, T. Han, Z. Gong, S. Samangooei, M. Monteiro, J. Menick, S. Borgeaud, A. Brock, A. Nematzadeh, S. Sharifzadeh, M. Binkowski, R. Barreira, O. Vinyals, A. Zisserman, and K. Simonyan. Flamingo: a visual language model for few-shot learning. 2022.

[3] P. Ammanabrolu, W. Cheung, W. Broniec, and M. O. Riedl. Automated storytelling via causal, commonsense plot ordering. In *AAAI Conference on Artificial Intelligence*, 2020.

[4] P. Ammanabrolu, J. Urbanek, M. Li, A. Szlam, T. Rocktäschel, and J. Weston. How to motivate your dragon: Teaching goal-driven agents to speak and act in fantasy worlds. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 807–833, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.64. URL `https://aclanthology.org/2021.naacl-main.64`.

[5] I. Armeni, Z. He, J. Gwak, A. Zamir, M. Fischer, J. Malik, and S. Savarese. 3D scene graph: A structure for unified semantics, 3D space, and camera. In *Intl. Conf. on Computer Vision (ICCV)*, pages 5664–5673, 2019.

[6] S. Badreddine, A. d'Avila Garcez, L. Serafini, and M. Spranger. Logic tensor networks. *Artificial Intelligence*, 303:103649, 2022.

[7] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *Intl. Conf. on Computer Vision (ICCV)*, 2019.

[8] B. Berger and P. W. Shor. Approximation alogorithms for the maximum acyclic subgraph problem. In *ACM-SIAM Symposium on Discrete Algorithms*, 1990.

[9] S. Black, L. Gao, P. Wang, C. Leahy, and S. Biderman. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow, Mar. 2021. URL `https://doi.org/10.5281/zenodo.5297715`. If you use this software, please cite it using these metadata.

[10] T. Bolukbasi, K.-W. Chang, J. Y. Zou, V. Saligrama, and A. T. Kalai. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL `https://proceedings.neurips.cc/paper/2016/file/a486cd07e4ac3d270571622f4f316ec5-Paper.pdf`.

[11] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, E. Brynjolfsson, S. Buch, D. Card, R. Castellon, N. Chatterji, A. Chen, K. Creel, J. Q. Davis, D. Demszky, C. Donahue, M. Doumbouya, E. Durmus, S. Ermon, J. Etchemendy, K. Ethayarajh, L. Fei-Fei, C. Finn, T. Gale, L. Gillespie, K. Goel, N. Goodman, S. Grossman, N. Guha, T. Hashimoto, P. Henderson, J. Hewitt, D. E. Ho, J. Hong, K. Hsu, J. Huang, T. Icard, S. Jain, D. Jurafsky, P. Kalluri, S. Karamcheti, G. Keeling, F. Khani, O. Khattab, P. W. Koh, M. Krass, R. Krishna, R. Kuditipudi, A. Kumar, F. Ladhak, M. Lee, T. Lee, J. Leskovec, I. Levent, X. L. Li, X. Li, T. Ma, A. Malik, C. D. Manning, S. Mirchandani, E. Mitchell, Z. Munyikwa, S. Nair, A. Narayan, D. Narayanan, B. Newman, A. Nie, J. C. Niebles, H. Nilforoshan, J. Nyarko, G. Ogut, L. Orr, I. Papadimitriou, J. S. Park, C. Piech, E. Portelance, C. Potts, A. Raghunathan, R. Reich, H. Ren, F. Rong, Y. Roohani, C. Ruiz, J. Ryan, C. Ré, D. Sadigh, S. Sagawa, K. Santhanam, A. Shih, K. Srinivasan, A. Tamkin, R. Taori, A. W. Thomas, F. Tramèr, R. E. Wang, W. Wang, B. Wu, J. Wu, Y. Wu, S. M. Xie, M. Yasunaga, J. You, M. Zaharia, M. Zhang, T. Zhang, X. Zhang, Y. Zhang, L. Zheng, K. Zhou, and P. Liang. On the opportunities and risks of foundation models, 2022.

[12] S. Bowman, N. Atanasov, K. Daniilidis, and G. Pappas. Probabilistic data association for semantic SLAM. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 1722–1729, 2017.

[13] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf`.

[14] A. Bucker, L. Figueredo, S. Haddadin, A. Kapoor, S. Ma, S. Vemprala, and R. Bonatti. Latte: Language trajectory transformer, 2022.

[15] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017.

[16] D. Chaves, J. Ruiz-Sarmiento, N. Petkov, and J. González-Jiménez. From object detection to room categorization in robotics. pages 1–6, 01 2020. doi: 10.1145/3378184.3378230.

[17] C. Chen, K. Lin, and D. Klein. Constructing taxonomies from pretrained language models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 2021.

[18] K. Choi, C. Cundy, S. Srivastava, and S. Ermon. LMPriors: Pre-trained language models as task-specific priors. In *NeurIPS 2022 Foundation Models for Decision Making Workshop*, 2022. URL https://openreview.net/forum?id=U2MnmJ7Sa4.

[19] N. Chomsky. *Aspects of the Theory of Syntax*. The MIT Press, Cambridge, 1965. URL http://www.amazon.com/Aspects-Theory-Syntax-Noam-Chomsky/dp/0262530074.

[20] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman. Training verifiers to solve math word problems, 2021.

[21] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. 3 edition, 2009.

[22] D. Dohan, W. Xu, A. Lewkowycz, J. Austin, D. Bieber, R. G. Lopes, Y. Wu, H. Michalewski, R. A. Saurous, J. Sohl-dickstein, K. Murphy, and C. Sutton. Language model cascades. In *International Conference on Machine Learning*, 2022.

[23] J. Dong, X. Fei, and S. Soatto. Visual-Inertial-Semantic scene representation for 3D object detection. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[24] D. Driess, F. Xia, M. S. M. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, W. Huang, Y. Chebotar, P. Sermanet, D. Duckworth, S. Levine, V. Vanhoucke, K. Hausman, M. Toussaint, K. Greff, A. Zeng, I. Mordatch, and P. Florence. Palm-e: An embodied multimodal language model, 2023.

[25] D. Fried, J.-B. Alayrac, P. Blunsom, C. Dyer, S. Clark, and A. Nematzadeh. Learning to segment actions from observation and narration. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*,

pages 2569–2588, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.231. URL https://aclanthology.org/2020.acl-main.231.

[26] C. Galindo, A. Saffiotti, S. Coradeschi, P. Buschka, J. Fernández-Madrigal, and J. González. Multi-hierarchical semantic maps for mobile robotics. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 3492–3497, 2005.

[27] N. D. Goodman and M. C. Frank. Pragmatic language interpretation as probabilistic inference. *Trends in Cognitive Sciences*, 20(11):818–829, 2016. ISSN 1364-6613. doi: https://doi.org/10.1016/j.tics.2016.08.005.

[28] H. P. Grice. Logic and conversation. In *Syntax and Semantics: Vol. 3: Speech Acts*, pages 41–58. Academic Press, 1975.

[29] M. Grinvald, F. Furrer, T. Novkovic, J. J. Chung, C. Cadena, R. Siegwart, and J. Nieto. Volumetric Instance-Aware Semantic Mapping and 3D Object Discovery. *IEEE Robotics and Automation Letters*, 4(3):3037–3044, 2019.

[30] Y. Gu, B. D. Mishra, and P. Clark. Do language models have coherent mental models of everyday things?, 2022.

[31] W. L. Hamilton., R. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems (NIPS)*, page 1025–1035, Dec. 2017.

[32] R. Hassin and S. Rubinstein. Approximations for the maximum acyclic subgraph problem. *Information Processing Letters*, 51(3):133–140, 1994.

[33] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. pages 770–778, 2016.

[34] T. M. Howard, S. Tellex, and N. Roy. A natural language planner interface for mobile manipulators. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6652–6659, 2014. doi: 10.1109/ICRA.2014.6907841.

[35] C. Huang, O. Mees, A. Zeng, and W. Burgard. Visual language maps for robot navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[36] N. Hughes, Y. Chang, and L. Carlone. Hydra: a real-time spatial perception engine for 3D scene graph construction and optimization. In *Robotics: Science and Systems (RSS)*, 2022. (pdf).

[37] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38, mar 2023.

[38] J. Jiang, L. Zheng, F. Luo, and Z. Zhang. Rednet: Residual encoder-decoder network for indoor rgb-d semantic segmentation, 2018. URL https://arxiv.org/abs/1806.01054.

[39] J. Johnson, A. Gupta, and L. Fei-Fei. Image generation from scene graphs. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[40] D. Kahneman. *Thinking, Fast and Slow*. New York, 2011.

[41] R. M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. 1972.

[42] J. Kerr, C. M. Kim, K. Goldberg, A. Kanazawa, and M. Tancik. Lerf: Language embedded radiance fields. 2023.

[43] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2017.

[44] T. Kollar, S. Tellex, D. K. Roy, and N. Roy. Toward understanding natural language directions. In *HRI 2010*, 2010.

[45] K. Kveraga, A. Ghuman, and M. Bar. Top-down predictions in the cognitive brain. *Brain and Cognition*, 65(2):145–168, 2007.

[46] O. Lang, Y. Gandelsman, M. Yarom, Y. Wald, G. Elidan, A. Hassidim, W. T. Freeman, P. Isola, A. Globerson, M. Irani, and I. Mosseri. Explaining in style: Training a gan to explain a classifier in stylespace. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 673–682, 2021.

[47] Y. T. Lee. Cse 421: Dynamic programming - longest path in a dag, 2018.

[48] A. K. Lew, M. H. Tessler, V. K. Mansinghka, and J. B. Tenenbaum. Leveraging unstructured statistical knowledge in a probabilistic language of thought. *Proceedings of the Annual Conference of the Cognitive Science Society*, 2020.

[49] B. Z. Li, W. Chen, P. Sharma, and J. Andreas. Lampp: Language models as probabilistic priors for perception and action. 2023.

[50] C. Li, H. Xiao, K. Tateno, F. Tombari, N. Navab, and G. D. Hager. Incremental scene understanding on dense SLAM. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 574–581, 2016.

[51] J. Li, D. Li, C. Xiong, and S. Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation, 2022.

[52] J. Li, D. Li, S. Savarese, and S. Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models, 2023.

[53] J. Lin, D. Fried, D. Klein, and A. Dragan. Inferring rewards from language in context. 2022.

[54] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. arXiv, 2019. doi: 10.48550/ARXIV.1907.11692. URL https://arxiv.org/abs/1907.11692.

[55] I. Loshchilov and F. Hutter. Decoupled weight decay regularization, 2019.

[56] C. Lu, R. Krishna, M. Bernstein, and F.-F. Li. Visual relationship detection with language priors. In *European Conference on Computer Vision*, pages 852–869, 2016.

[57] H. Luo, A. Yue, Z.-W. Hong, and P. Agrawal. Stubborn: A strong baseline for indoor object navigation, 2022.

[58] C. Lynch and P. Sermanet. Language conditioned imitation learning over unstructured data. arXiv, 2020. doi: 10.48550/ARXIV.2005.07648. URL https://arxiv.org/abs/2005.07648.

[59] Manolis Savva*, Abhishek Kadian*, Oleksandr Maksymets*, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra. Habitat: A Platform for Embodied AI Research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[60] C. Matuszek, E. Herbst, L. Zettlemoyer, and D. Fox. *Learning to Parse Natural Language Commands to a Robot Control System*, pages 403–415. Springer International Publishing, Heidelberg, 2013. ISBN 978-3-319-00065-7. doi: 10.1007/978-3-319-00065-7_28. URL https://doi.org/10.1007/978-3-319-00065-7_28.

[61] J. McCormac, R. Clark, M. Bloesch, A. Davison, and S. Leutenegger. Fusion++: Volumetric object-level SLAM. In *Intl. Conf. on 3D Vision (3DV)*, pages 32–41, 2018.

[62] G. A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11): 39–41, Nov. 1995. ISSN 0001-0782. doi: 10.1145/219717.219748. URL https://doi.org/10.1145/219717.219748.

[63] J. Mirault, J. Snell, and J. Grainger. You that read wrong again! a transposed-word effect in grammaticality judgments. *Psychological Science*, 29:095679761880629, 10 2018. doi: 10.1177/0956797618806296.

[64] B. B. Murdock. The serial position effect of free recall. *Journal of Experimental Psychology*, 64:482–488, 1962.

[65] S. Nair, E. Mitchell, K. Chen, B. Ichter, S. Savarese, and C. Finn. Learning language-conditioned robot behavior from offline data and crowd-sourced annotation, 2021.

[66] L. Nicholson, M. Milford, and N. Sünderhauf. QuadricSLAM: Dual quadrics from object detections as landmarks in object-oriented SLAM. *IEEE Robotics and Automation Letters*, 4:1–8, 2018.

[67] K. Ok, K. Liu, and N. Roy. Hierarchical object map estimation for efficient and robust navigation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1132–1139, 2021. doi: 10.1109/ICRA48506.2021.9561225.

[68] OpenAI. Introducing chatgpt. https://openai.com/blog/chatgpt, 2022.

[69] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. Lowe. Training language models to follow instructions with human feedback, 2022.

[70] C. Painter. *Learning Through Language in Early Childhood*. Continuum Collection. Bloomsbury Publishing, 2005. ISBN 9781847143945. URL `https://books.google.com/books?id=4sB0i-DfT0MC`.

[71] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. 2019.

[72] 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. arXiv, 2021. doi: 10.48550/ARXIV.2103.00020. URL `https://arxiv.org/abs/2103.00020`.

[73] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL `http://jmlr.org/papers/v21/20-074.html`.

[74] S. K. Ramakrishnan, A. Gokaslan, E. Wijmans, O. Maksymets, A. Clegg, J. M. Turner, E. Undersander, W. Galuba, A. Westbury, A. X. Chang, M. Savva, Y. Zhao, and D. Batra. Habitat-matterport 3d dataset (HM3d): 1000 large-scale 3d environments for embodied AI. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL `https://openreview.net/forum?id=-v4OuqNs5P`.

[75] A. Rosinol, A. Gupta, M. Abate, J. Shi, and L. Carlone. 3D dynamic scene graphs: Actionable spatial perception with places, objects, and humans. In *Robotics: Science and Systems (RSS)*, 2020. doi: 10.15607/RSS.2020.XVI.079. URL `http://news.mit.edu/2020/robots-spatial-perception-0715`. (pdf), (media), (video).

[76] A. Rosinol, A. Violette, M. Abate, N. Hughes, Y. Chang, J. Shi, A. Gupta, and L. Carlone. Kimera: from SLAM to spatial perception with 3D dynamic scene graphs. *Intl. J. of Robotics Research*, 40(12–14):1510–1546, 2021. arXiv preprint: 2101.06894, (pdf).

[77] R. Rosu, J. Quenzel, and S. Behnke. Semi-supervised semantic mapping through label propagation with semantic texture meshes. *Intl. J. of Computer Vision*, 06 2019.

[78] J.-R. Ruiz-Sarmiento, C. Galindo, and J. Gonzalez-Jimenez. Building multiversal semantic maps for mobile robot operation. *Knowledge-Based Systems*, 119:257–272, 2017.

[79] N. M. M. Shafiullah, C. Paxton, L. Pinto, S. Chintala, and A. Szlam. Clip-fields: Weakly supervised semantic fields for robotic memory, 2022.

[80] D. Shah, B. Osinski, B. Ichter, and S. Levine. Robotic Navigation with Large Pre-Trained Models of Language, Vision, and Action. 2022.

[81] C. E. Shannon. Prediction and entropy of printed english. *The Bell System Technical Journal*, 30(1):50–64, 1951. doi: 10.1002/j.1538-7305.1951.tb01366.x.

[82] P. Sharma, B. Sundaralingam, V. Blukis, C. Paxton, T. Hermans, A. Torralba, J. Andreas, and D. Fox. Correcting robot plans with natural language feedback. arXiv, 2022. doi: 10.48550/ARXIV.2204.05186. URL https://arxiv.org/abs/2204.05186.

[83] M. Shridhar, L. Manuelli, and D. Fox. Cliport: What and where pathways for robotic manipulation. arXiv, 2021. doi: 10.48550/ARXIV.2109.12098. URL https://arxiv.org/abs/2109.12098.

[84] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *European Conf. on Computer Vision (ECCV)*, 2012.

[85] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg. Progprompt: Generating situated robot task plans using large language models. In *Second Workshop on Language and Reinforcement Learning*, 2022. URL https://openreview.net/forum?id=aflRdmGOhw1.

[86] S. Song, S. Lichtenberg, and J. Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 567–576. IEEE Computer Society, 2015. doi: 10.1109/CVPR.2015.7298655. URL https://doi.ieeecomputersociety.org/10.1109/CVPR.2015.7298655.

[87] R. Speer, J. Chin, and C. Havasi. ConceptNet 5.5: an open multilingual graph of general knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 4444–4451. AAAI Press, 2017.

[88] T. R. Sumers, R. D. Hawkins, M. K. Ho, T. L. Griffiths, and D. Hadfield-Menell. Linguistic communication as (inverse) reward design, 2022.

[89] Z. Sun, Y. Shen, Q. Zhou, H. Zhang, Z. Chen, D. Cox, Y. Yang, and C. Gan. Principle-driven self-alignment of language models from scratch with minimal human supervision, 2023.

[90] A. Szot, A. Clegg, E. Undersander, E. Wijmans, Y. Zhao, J. Turner, N. Maestre, M. Mukadam, D. Chaplot, O. Maksymets, A. Gokaslan, V. Vondrus, S. Dharur, F. Meier, W. Galuba, A. Chang, Z. Kira, V. Koltun, J. Malik, M. Savva, and D. Batra. Habitat 2.0: Training home assistants to rearrange their habitat. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[91] A. Talmor, O. Yoran, R. L. Bras, C. Bhagavatula, Y. Goldberg, Y. Choi, and J. Berant. CommonsenseQA 2.0: Exposing the limits of AI through gamification. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021. URL https://openreview.net/forum?id=qF7FlUT5dxa.

[92] S. Tellex, T. Kollar, S. Dickerson, M. R. Walter, A. G. Banerjee, S. J. Teller, and N. Roy. Understanding natural language commands for robotic navigation and mobile manipulation. In *AAAI*, 2011. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI11/paper/view/3623.

[93] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems (NIPS)*, volume 30. Curran Associates, Inc., 2017.

[94] W. von Humboldt, P. Heath, and H. Aarsleff. *On Language: The Diversity of Human Language-Structure and its Influence on the Mental Development of Mankind*. Texts in German Philosophy. Cambridge University Press, 1988. ISBN 9780521315135. URL https://books.google.com/books?id=0CXEQwAACAAJ.

[95] B. Wang and A. Komatsuzaki. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax, May 2021.

[96] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, W. Liu, and B. Xiao. Deep high-resolution representation learning for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(10):3349–3364, 2021. doi: 10.1109/TPAMI.2020.2983686.

[97] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou. Chain of thought prompting elicits reasoning in large language models. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=_VjQlMeSB_J.

[98] K. Yadav, S. K. Ramakrishnan, J. Turner, A. Gokaslan, O. Maksymets, R. Jain, R. Ramrakhya, A. X. Chang, A. Clegg, M. Savva, E. Undersander, D. S. Chaplot, and D. Batra. Habitat challenge 2022. https://aihabitat.org/challenge/2022/, 2022.

[99] X. Ye and G. Durrett. The unreliability of explanations in few-shot prompting for textual reasoning. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[100] E. Zelikman, Y. Wu, J. Mu, and N. D. Goodman. STar: Bootstrapping reasoning with reasoning. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[101] A. Zeng, M. Attarian, B. Ichter, K. Choromanski, A. Wong, S. Welker, F. Tombari, A. Purohit, M. Ryoo, V. Sindhwani, J. Lee, V. Vanhoucke, and P. Florence. Socratic models: Composing zero-shot multimodal reasoning with language. In *Submitted to The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=G2Q2Mh3avow`. under review.

[102] D. Zhukov, J.-B. Alayrac, R. G. Cinbis, D. Fouhey, I. Laptev, and J. Sivic. Cross-task weakly supervised learning from instructional videos. In *Computer Vision and Pattern Recognition*, 2019.