Navigation with Large Language Models: Semantic Guesswork as a Heuristic for Planning

Anonymous Author(s)
Affiliation
Address
email

Abstract: Navigation in unfamiliar environments presents a major challenge for robots: while mapping and planning techniques can be used to build up a representation of the world, quickly discovering a path to a desired goal in unfamiliar settings with such methods often requires lengthy mapping and exploration. Humans can rapidly navigate new environments, particularly indoor environments that are laid out logically, by leveraging semantics – for example, we know that a kitchen often adjoins a living room, or that an exit sign indicates the way out. Language models can allow robots to acquire such semantic knowledge, but directly using language models to instruct a robot how to reach some destination can also be impractical – while language models might produce a narrative about how to reach some goal, because they are not grounded in real-world observations, this narrative might be arbitrarily wrong. Therefore, in this paper we study how to utilize the "semantic guesswork" produced by language models as a guiding heuristic for planning algorithms. In this way, our method, Language Frontier Guide (LFG), uses the language model to bias planning and exploration, but does not rely on the language model to always correctly determine the path to the goal. We propose specific approaches to use language model proposals as heuristics, and evaluate our method on simulated indoor navigation benchmarks, showing that our approach outperforms uninformed exploration and other ways of using language models. For more results please visit our project website https://sites.google.com/view/lfg-anon/home

Keywords: navigation, language models, planning, semantics

1 Introduction

Navigation in complex open-world environments is conventionally viewed as the largely geometric problem of determining collision-free paths that traverse the environment from one location to another. However, real-world environments possess *semantics*. Imagine navigating an airport to get to a terminal — your prior knowledge about the way such buildings are constructed provides extensive guidance, even if this particular airport is unfamiliar to you. Language models and various language embedding techniques have been studied extensively as ways to interpret the semantics in user-specified *instructions* (e.g., parsing "go to the television in the living room" and grounding it in a specific spatial location), and such models can provide much more assistance in robotic navigation scenarios by capturing rich semantic knowledge about the world. A language model might be able to propose that the living room is likely adjacent to a kitchen, and is less likely to be adjacent to the garden shed. However, since language models are not *grounded* in the real world, such models do not know the spatial layout of the robot's surroundings. To utilize the semantic knowledge in language models to aid in embodied tasks, we should not just blindly *follow* the language model suggestions, but instead use them as proposals or navigational heuristics. In this paper, we study how that might be accomplished.

We study this idea in the context of embodied visual navigation, where a robot is tasked with reaching a goal denoted by a natural language query q (see Fig. 1) in a *novel* environment, and has access to visual observations from an onboard camera. Note that the robot has no prior experience in the

Submitted to the 7th Conference on Robot Learning (CoRL 2023). Do not distribute.

Figure 1: In constrast to methods that use LLM plans directly, Language Frontier Guide (LFG) uses a language model to *score* subgoal candidates, and uses these scores to guide a heuristic-based planner.

target environment, and must explore the environment to look for the goal. We posit that an effective way to build such a robotic system is to combine the high-level reasoning capabilities of pre-trained LLMs with state-of-the-art robot navigation stacks, that are adept at collision avoidance, learning navigational affordances, and efficiently exploring unknown environments. While plans generated by an LLM may not be sufficient for navigation by themselves, they contain powerful cues that can be used to *inform* or *guide* the behavior of the underlying navigation stack for the language navigation task. We show that this idea can be combined with frontier-based exploration, where the agent maintains a set of unvisited locations at its frontier, and the LLM "scores" these frontiers according to their likelihood of reaching the goal q.

We propose Language Frontier Guide, or LFG, a method for leveraging the reasoning capabilities of LLMs to produce a *search heuristic* for guiding exploration of previously unseen environments, combining the strengths of search-based planning, and LLM reasoning. We show that our proposed approach can be combined with both a geometric navigation pipeline, building a metric map of the environment for planning, and using a hand-designed controller, yielding a simple system for navigating to open-vocabulary natural language goals. Our experiments show that LFG can identify and predict simple patterns in previously unseen environments to accelerate goal-directed exploration. Furthermore, we also report that LFG forms a simple, yet performant, baseline on the Habitat ObjectNav benchmark, matching the state-of-the-art in learned navigation and outperforming LLM-based approaches, all while requiring no task-specific training or fine-tuning. We have released our prompts, scoring algorithm, and evaluation code, please see supplemental material.

2 Related Work

Vision-Based Navigation Navigation is conventionally approached as a largely geometric problem, where the aim is to map an environment and use that map to find a path to a goal location [1]. Learning-based approaches can exploit patterns in the training environments, particularly by learning vision-based navigation strategies through reinforcement or imitation [2–6]. Somewhat similar to our work, PONI [6] uses a learned potential function to prioritize frontier points to explore, while in our work, we use a language model to rank frontier points. Notably, those methods do not benefit from prior semantic knowledge (e.g., from the web), and must rely entirely on patterns discovered from offline or online navigational data. Our aim is specifically to bring semantic knowledge into navigation, to enable robots to more effectively search for a goal in a new environment.

Semantic Knowledge-Guided Navigation Prior knowledge about the semantics of indoor environments can provide significantly richer guidance. With the advent of effective open-vocabulary vision models [7, 8], some works have recently explored incorporating their semantic knowledge into models for navigation and other robotic tasks with the express aim of improving performance at *instruction following* [9–13]. In general within robotics, such methods have either utilized pre-trained vision-language representations [14–16], or used language models directly to make decisions [17–22]. Our aim is somewhat different: while we also focus on language-specified goals, we are primarily concerned with utilizing the semantics in pre-trained language models to help a robot figure out how to actually reach the goal, rather than utilizing the language models to more effectively interpret a language instruction. While language models can output reasonable substeps for temporally extended tasks in some settings [23, 24], there is contradictory evidence about their ability to actually plan [25], and because they are unaware of the observations and layout in a particular environment, their "plans" depend entirely on the context that is provided to them. In contrast

to prior work, our approach does not rely on the language model producing a *good* plan, but merely a heuristic that can bias a dedicated planner to reach a goal more effectively. In this way, we use the language models more to produce *suggestions* rather than actual plans.

Language Model-Guided Navigation Some works have sought to combine predictions from language models with either planning or probabilistic inference [13, 26], so as to not rely entirely on forward prediction from the language model to take actions. However, these methods are more aimed at filtering out *infeasible* decisions, for example by disallowing actions that a robot is incapable of performing, and still focus largely on being able to interpret and process instructions, rather than using the language model as a source of semantic hints. In contrast, by incorporating language model suggestions as heuristics into a heuristic planner, our approach can completely override the language model predictions if they are incorrect, while still making use of them if they point the way to the goal.

Another branch of recent research [27–29] has taken a different approach to ground language models, by making it possible for them to read in image observations directly. While this represents a promising alternative approach to make language models more useful for embodied decision making, we believe it is largely orthogonal and complementary to our work: although vision-language models can produce more grounded inferences about the actions a robot should take, they are still limited only to *guessing* when placed in unfamiliar environments. Therefore, although we use ungrounded language-only models in our evaluation, we expect that our method could be combined with vision-language models easily, and would provide complementary benefits.

3 Problem Formulation and Overview

Our objective is to design a high-level planner that takes as input a natural language query q (e.g., "find the bedside table"), explores the environment in search of the queried object, and commands a low-level policy to control a robotic agent. To do this, we maintain an episodic memory of the environment $\mathcal{M}$ in the form of a 2D map of the environment, where grid cells contain information about (i) occupancy, and (ii) semantic labels of the objects occupying them. The simplest way to solve this task is Frontier-Based Exploration (FBE) [30], where a robot maintains a set of unexplored *frontiers* in it's memory, and explores randomly to reach the goal. *Can we do better with access to large language models (LLMs)?*

At the core of LFG is the task of *ranking* subgoal proposals. Formally, let's assume we have the task by query q, a partially explored environment stored in $\mathcal{M}$, and a set $\mathcal{S}$ of n subgoal proposals $s_1, s_2, \ldots, s_n$ (e.g., "go to the kitchen", "go to the bedroom", etc.). Our goal is to score these subgoal proposals with $p(s_i, q, \mathcal{M})$, the probability that the candidate $s_i \in \mathcal{S}$ will lead to the goal q given the current state of the environment, described through $\mathcal{M}$.

We posit that we can leverage the semantic reasoning capabilities of large language models (LLMs) to parse the episodic memory $\mathcal{M}$ and extract information about the most relevant subgoals to satisfy the goal query. However, rather than directly using LLM scores for planning [22, 24], we incorporate them as a goal-directed *heuristic* to inform the search process. This has two distinct advantages: (i) when the LLM is right, it nudges the search towards the goal, and when it is wrong (or uncertain), we can still default to the underlying FBE algorithm, allowing recovery from LLM failures, and (ii) it allows us to combine the signal from LLMs with other scores that may be more grounded, e.g. distance to subgoals, making the system more versatile.

4 LFG: Scoring Subgoals by Polling LLMs

Current approaches to using LLMs for robotic planning rely either on direct language or code generation from the LLM [19, 23], or use the log likelihoods (or *logprobs*) provided by the LLM to score subgoal proposals, conditioned on a task-relevant prompt [13, 22, 24, 26]. While logprobs capture some form of a score across subgoals — they denote the probability of subgoal tokens in autoregressive prediction — they do not represent the desired task-relevant probability $p(s_i, q, \mathcal{M})$, and fail to assign similar scores, say, to different subgoals that are semantically similar but have different tokenizations (see our experiments in Section 6 for a comparison). Furthermore, most ca-

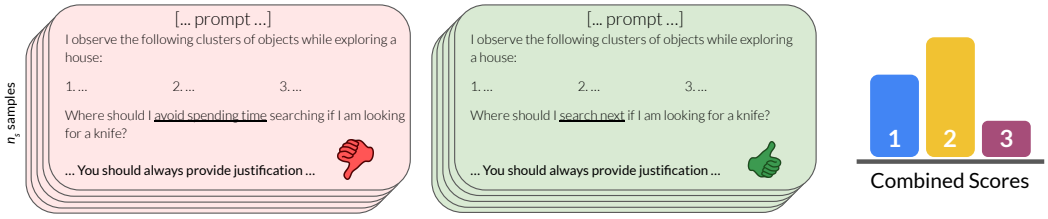


Figure 2: LFG scores subgoals with an empirical estimate of the likelihoods by sampling an LLM n_s times with both positive and negative prompts, and uses chain-of-thought to obtain reliable scores. These scores are used by a high-level planner as *heuristics* to guide search. For full prompts, see Appendix C.

pable LLMs of today are available through APIs that do not expose the ability to query logprobs.[1] And lastly, even if reliable logprobs were available, they are incompatible with chain-of-thought prompting [31], which we find to be crucial to success in spatial reasoning.

To overcome these challenges, LFG uses a novel approach to extract task-relevant likelihoods from LLMs. Instead of querying the LLM for logprobs directly, we *poll* the LLMs by sampling the most likely subgoal n_s times, conditioned on a task-relevant prompt. We then use these samples to empirically estimate the likelihood of each subgoal. To get informative and robust likelihood estimates, we use a chain-of-thought prompting (CoT) technique [31], to improve the quality and interpretability of the scores, and use a combination of positive and negative prompts to gather unbiased likelihood estimates. Figure 2 outlines our scoring technique, with the full prompt provided in Appendix C. We now describe the details of our scoring technique.

Structured query: We rely on in-context learning by providing an example of a structured query-response pair to the LLM, and ask it to pick the most likely subgoal that satisfies the query. To sample a subgoal from $\mathcal{S}$ using a language model, we prompt it to generate a structured response, ending with ```Answer: i''`. This structure allows us to always sample a *valid* subgoal, without having to ground LLM generations in the environment [23].

Positives and negatives: We find that only using positive prompts (e.g., "which subgoal is most likely to reach the goal") leads to likelihood estimates being uninformative for cases where the LLM is not confident about any subgoal. To overcome this, we also use negative prompts (e.g., "which subgoal is least likely to be relevant for the goal"), which allows us to score subgoals by eliminating/downweighting subgoals that are clearly irrelevant. We then use the difference between the positive and negative likelihoods to rank subgoals.

Chain-of-thought prompting: A crucial component of getting interpretable and reliable likelihood estimates is to encourage the LLM to *justify* its choice by chain-of-thought prompting. As demonstrated in prior works, CoT elicits interpretability and reasoning capabilities in LLMs, and while we don't explicitly use the generated reasonings in our approach (great future work direction), we find that CoT improves the quality and consistency of the likelihood estimates. Additionally, it also helps maintain interpretability, to better understand why the LFG-equipped agent takes certain decisions.

Algorithm 1: Scoring Subgoals with LFG

Data: Subgoal descriptors $\{l_i \forall s_i \in \mathcal{S}\}$
pPrompt $\leftarrow$ PositivePrompt($\{l_i\}$)
nPrompt $\leftarrow$ NegativePrompt($\{l_i\}$)
pSamples $\leftarrow \phi$
nSamples $\leftarrow \phi$
for n_s **do**
  pSamples.append(sampleLLM(pPrompt))
  nSamples.append(sampleLLM(nPrompt))
pScores $\leftarrow$ sum(pSamples) / n_s
nScores $\leftarrow$ sum(nSamples) / n_s
return pScores, nScores

In summary, we score subgoals by sampling the LLM multiple times and empirically estimating the likelihood of each subgoal. We use a combination of positive and negative prompts to get unbiased likelihood estimates, and use chain-of-thought prompting to improve the quality and interpretability of the scores (Figure 2). We will now discuss how these scores can be incorporated into a navigation system as *search heuristics*.

[1] Most notably, OpenAI's Chat API for GPT-3.5 and GPT-4, Google's PaLM API, and Anthropic's Claude API all do not support logprobs.

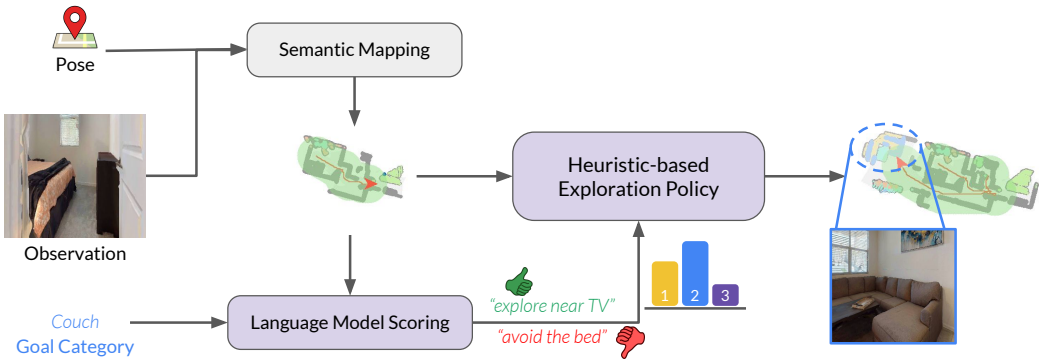


Figure 3: Overview of LFG for object goal navigation. Based on the pose and observations, the method builds a semantic map and obstacle map. The heuristic-based exploration policy scores nearby object clusters using large language models to rank which of the clusters adheres best to the goal category. Navigation to the subgoal is completed by a low-level policy.

5 LLM Heuristics for Goal-Directed Exploration

Given the LLM scoring pipeline outlined in the previous section, our key insight is that we can incorporate these scores in a search-based planning pipeline to *heuristically* guide the search process. We instantiate LFG using an FBE search algorithm and LLM scores generated via polling.

FBE: This method maintains a map of the seen parts of the environment, and a frontier separating it from the unexplored parts. By navigating to the *nearest* point of the frontier, the robot explores new areas of the environment until it finds the goal object or completes exploration without finding it. A standard FBE implementation is presented in Algorithm 2 in black text. The agent maintains a 2D metric map of its surroundings, with each grid cell containing information about occupancy and semantic labels. In our analysis, we use ground-truth semantic labels available from the simulator to isolate the planning problem from perception challenges.

At a fixed re-planning rate, the high-level planner computes its frontier f_i (Line 10), and picks the frontier point that is *closest* to the current location, i.e., maximizing the distance score (Line 17), and then navigates to this frontier (Line 22). At any point in this process, if the agent's semantic detector detects an object of the same category as the query q, it navigates directly to this object and the trajectory ends.

Incorporating LLM scores: Our method, LFG, extends FBE by using an additional search heuristic obtained by polling LLMs for their opinions. The modifications to FBE are marked in purple. After enumerating frontiers, LFG uses its semantic map to parse object clusters at each frontier f_i (Line 12). These object clusters are converted into textual strings, and form the subgoal candidates that can be scored using Algorithm 1. We associate each frontier point f_i with the nearest object cluster c_i (Line 18), and compute LLM scores for each point as follows:

$$h(f_i, q) = w_p \cdot \text{LLM}_{\text{pos}}(c_i) - w_n \cdot \text{LLM}_{\text{neg}}(c_i) - \text{dist}(f_i, p), \tag{1}$$

where p is the current position of the agent, and w_p, w_n are hyperparameters (see Appendix A). We then choose the frontier with the highest score to be the next subgoal (Line 22), navigate to it using a fast marching method-based local controller, and repeat the planning process. Algorithm 2 outlines the full system for using polling-based LLM scores as a planning heuristic. For more details of the 'parseObjects', 'clusterObjects' and 'ScoreSubgoals' function, please refer to the suppmentary website or code[2]. Figure 4 shows an example of subgoal selection with and without a language score.

6 System Evaluation

We now evaluate the performance of LFG for the task of goal-directed exploration in a photorealistic simulator. Our experiments show that LFG outperforms existing LLM-based exploration algorithms

[2]https://sites.google.com/view/lfg-anon/home

Algorithm 2: Instantiating LFG for ObjectGoal Navigation

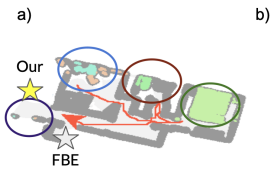
```
Data: o_0, Goal language query q
subgoal ← None
while not done do
    o_t ← getObservation()
    obstacleMap, semanticMap ← mappingModule(o_t[depth], o_t[semantic])
    if q in semanticMap then
        subGoal ← getLocation(semanticMap, q)
    else
        if numSteps % τ == 0 then
            // replanning
            location ← getCurrentLocation()
            frontier ← getFrontier(obstacleMap)
            objects ← parseObjects(semanticMap)
            objectClusters ← clusterObjects(objects)
            LLM_pos, LLM_neg ← ScoreSubgoals(objectClusters)
            scores ← []
            for point in frontier do
                distance ← distTo(location, point)
                scores[point] ← – distance
                closestCluster ← getClosestCluster(objectClusters, point)
                i ← clusterID(closestCluster)
                if dist(closestCluster, point) < δ then
                    // incorporate language scores
                    scores[point] ← w_p · LLM_pos [i] - w_n · LLM_neg [i] - distance
            subgoal ← argmax(scores)
    numSteps ← numSteps +1
    goTo(subGoal)
```

by a wide margin on the ObjectNav task, owing to the high quality scores incorporated as search heuristics. Furthermore, we also report that under the assumptions of ground-truth semantics, LFG outperforms several learned baselines and closely matches the state-of-the-art, all without requiring any training data and solely relying on heuristics that tap into the semantic information stored in pre-trained LLMs.

6.1 Benchmark Results

6.2 Setup and Baselines

We evaluate LFG for the task of object-goal navigation in the Habitat simulator [32], where the agent is placed into a simulated environment with photo-realistic graphics, and is tasked with finding a query object from one of 10 categories (e.g., "toilet", "bed", "couch" etc.). Following the setup of the 2022 Habitat ObjectNav Challenge [33], the agents are allowed to train on 135 training scenes, and are evaluated on 35 scenes from the the Habitat-Matterport3D (HM3D) Semantics v0.1 dataset [34]. We run 10 evaluation episodes per scene and report two metrics: the average success rate, and success weighted by optimal path length (SPL), the default metrics for the benchmark. We emphasize that *LFG requires no training*, and we do not use the training scenes from HM3D.



b)

Object clusters	Language Scores	
	Positive	Negative
1) chair, table	0.8	0.2
2) chair, table	0.8	0.2
3) bed	0.3	0.6
4) toilet	0.0	1.0

Figure 4: Examples of object clusters scored by LFG and FBE. LFG discourages the agent from entering the room (with bed and toilet), arriving at goal faster.

Figure 5: Qualitative example of a negative score influencing the agent's decision. LFG discourages the agent from exploring the bedroom and living room, leading to fast convergence toward the goal, whereas FBE fails. The CoT reasoning given by the LLM is shown in purple, justifying its score.

Figure 6: Tolerance to LLM failures. An example rollout of LFG compensating for LLM failure. FBE takes over in this case and eventually succeeds, whereas the Greedy agent fails.

We compare to three classes of baselines: (i) learning-based baselines that learn navigation behavior from demonstrations or online experience in the training scenes, either using reinforcement learning [35], imitation learning [36, 37], or both [38], on anywhere between 12M-2B frames of experience, (ii) search-based baselines [30, 39], and (iii) LLM-based baselines that do not use the training data directly, but leverage the semantic knowledge inside foundation models to guide embodied tasks [17, 40].

Evaluating LFG on the HM3D benchmark, we find that it significantly outperforms search and LLM-based baselines (Table 1), maxing out the benchmark (note that the highest reported success rate on the benchmark is 70% [38]). Greedy LLM struggles on the task due to several LLM planning failures, causing the episodes to fail. LFG significantly outperforms the vanilla FBE baseline by leveraging semantic priors from LLMs to score subgoals intelligently (Figure 4). Comparing to learning-based baselines, we find that LFG outperforms most of them and closely matches the state-of-the-art on the task, proving the competence of our polling and heuristic approach.

L3MVN [40], which uses a combination of LLMs and search, performs slightly better than FBE, but is unable to fully leverage the semantics in LLMs. While being close in spirit to LFG, it suffers from some key limitations: (i) it uses a tiny language model (GPT-2 for open-sourced version), which likely does not contain strong semantic priors for the agent to leverage, and (ii) it uses a simple logprob-like scoring scheme, which we show below is not very effective. Another closely related work, LGX [17], uses a variant of greedy LLM scoring, and hence fails to perform reliably on the benchmark.

Probing deeper into the strong performance of LFG, we ablated various components of our scoring pipeline and studied the change in performance. Note that LGX (Greedy) and L3MVN (No CoT, Logprobs) can be seen as ablations of LFG. Table 2 shows that modifying both the prompting and scoring mechanisms used by LFG lead to large drops in performance. Most notably, scoring via polling (+7.8%) and CoT (+6.6%) are both essential to the strong performance of LFG. Further-

Method	Success	SPL	Training Data
DD-PPO [35]	27.9	14.2	2.5B
Habitat-Web [36]	57.6	23.8	12M
OVRL-v2 [37]	64.7	28.1	12M
PIRLNav [38]	**70.4**	**34.1**	37M
FBE$^{\pi}$	61.1	34.0	0
SemExp [39]	63.1	0.29	10M
Greedy LLM$^{\pi}$ [17]	54.4	26.9	**0**
L3MVN$^{\pi}$ [40]	62.4		**0**
LFG (Ours)$^{\pi}$	**68.9**	**36.0**	**0**

Table 1: LFG outperforms all LLM-based baselines on HM3D ObjectNav benchmark, and can achieve close to SOTA performance without any pre-training. $^{\pi}$These baselines were evaluated with ground-truth semantics.

Method	Success	Δ
LFG (Full)	**68.9**	–
Greedy	54.4	(14.5)
Prompting		
No CoT	62.3	(6.6)
Only Positives	64.2	(4.7)
Only Negatives	63.9	(5.0)
Scoring		
Random	61.1	(7.8)
Logprobs	60.4	(8.5)
Ask	62.4	(6.5)

Table 2: Comparing LFG to various prompting and scoring schemes for LLM-guided planning. We find that CoT prompting with positives and negatives, combined with polling, are essential to achieve the best performance.

more, we find that using only positive prompts also hurts performance (-4.7%). Popular approaches for using LLMs for planning — Greedy (-14.5%) and Logprobs (-8.5%) — are significantly outperformed by LFG.

Qualitative analysis: To demonstrate how LFG effectively uses LLMs, we visualize some qualitative examples of successful episodes. Figure 6 shows an example where the negative prompts used as part of LFG discourage the agent from exploring 3 out of 4 frontiers, accelerating the exploration process, and ending in success. FBE takes $2\times$ longer in this situation. Figure 6 highlights an important capability of LFG: falling back to search when the LLM is uncertain or wrong. In this case, the LLM makes a mistake and leads to a dead-end. While a Greedy planner would fail at this task, LFG succeeds by defaulting to the underlying FBE module.

7 Discussion

We presented LFG, a method for utilizing language models for *semantic guesswork* to help navigate to goals in new and unfamiliar environments. The central idea in our work is that, while language models can bring to bear rich semantic understanding, their ungrounded inferences about how to perform navigational tasks are better used as suggestions and heuristics rather than plans. We formulate a way to derive a heuristic score from language models that we can then incorporate into a planning algorithm, and use this heuristic planner to reach goals in new environments more effectively. This way of using language models benefits from their inferences when they are correct, and reverts to a more conventional unguided search when they are not.

Limitations and future work: While our experiments provide a validation of our key hypothesis, they have a number of limitations. First, we largely abstract away the problem of deriving semantic representations of the robot's visual observation by using the ground truth semantics provided by our simulator. A realistic deployment of this system would use a separate perception module (e.g., an image captioner or vision-language model) to provide the descriptions that would be fed into the language model. We believe that despite this limitation, our experiments are still a good way to study the central question, since the right way of using language models should not depend strongly on how the image annotations are actually produced. However, addressing this limitation will be crucial to turn our prototype into a system suitable for the real world. Second, our evaluation uses simulated indoor environments, and while prior work has shown that policies trained in these simulators do transfer to real-world settings [41], the role of semantics in navigation likely differs drastically across domains – e.g., navigating a forest might implicate semantics very differently than navigating an apartment building. We believe that the "no training required" feature of LFG makes it a viable candidate for sim-to-real transfer in simple environments, when coupled with a suitable semantic model [42–44]. Exploring the applicability of semantics derived from language models in other settings would be another promising and exciting direction for future work.

References

[1] G. N. DeSouza and A. C. Kak. Vision for mobile robot navigation: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 24(2):237–267, 2002. 2

[2] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, and A. van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3674–3683, 2018. 2

[3] N. Savinov, A. Dosovitskiy, and V. Koltun. Semi-parametric topological memory for navigation. *arXiv preprint arXiv:1803.00653*, 2018.

[4] N. Hirose, F. Xia, R. Martín-Martín, A. Sadeghian, and S. Savarese. Deep visual MPC-policy learning for navigation. *IEEE Robotics and Automation Letters*, 2019.

[5] D. Shah, A. Sridhar, A. Bhorkar, N. Hirose, and S. Levine. GNM: A General Navigation Model to Drive Any Robot. In *arXiV*, 2022. URL https://arxiv.org/abs/2210.03370.

[6] S. K. Ramakrishnan, D. S. Chaplot, Z. Al-Halah, J. Malik, and K. Grauman. Poni: Potential functions for objectgoal navigation with interaction-free learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18890–18900, 2022. 2

[7] X. Chen, X. Wang, S. Changpinyo, A. Piergiovanni, P. Padlewski, D. Salz, S. Goodman, A. Grycner, B. Mustafa, L. Beyer, et al. Pali: A jointly-scaled multilingual language-image model. *arXiv preprint arXiv:2209.06794*, 2022. 2

[8] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 2021. 2

[9] A. Majumdar, A. Shrivastava, S. Lee, P. Anderson, D. Parikh, and D. Batra. Improving vision-and-language navigation with image-text pairs from the web. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VI 16*, pages 259–274. Springer, 2020. 2

[10] A. Majumdar, G. Aggarwal, B. Devnani, J. Hoffman, and D. Batra. Zson: Zero-shot object-goal navigation using multimodal goal embeddings. *arXiv preprint arXiv:2206.12403*, 2022.

[11] B. Chen, F. Xia, B. Ichter, K. Rao, K. Gopalakrishnan, M. S. Ryoo, A. Stone, and D. Kappler. Open-vocabulary queryable scene representations for real world planning. *arXiv preprint arXiv:2209.09874*, 2022.

[12] C. Huang, O. Mees, A. Zeng, and W. Burgard. Visual language maps for robot navigation. *arXiv preprint arXiv:2210.05714*, 2022.

[13] D. Shah, B. Osinski, B. Ichter, and S. Levine. LM-nav: Robotic navigation with large pre-trained models of language, vision, and action. In *Annual Conference on Robot Learning (CoRL)*, 2022. 2, 3

[14] M. Shridhar, L. Manuelli, and D. Fox. Cliport: What and where pathways for robotic manipulation. In *Proceedings of the 5th Conference on Robot Learning (CoRL)*, 2021. 2

[15] N. M. M. Shafiullah, C. Paxton, L. Pinto, S. Chintala, and A. Szlam. Clip-fields: Weakly supervised semantic fields for robotic memory, 2023.

[16] K. Jatavallabhula, A. Kuwajerwala, Q. Gu, M. Omama, T. Chen, S. Li, G. Iyer, S. Saryazdi, N. Keetha, A. Tewari, J. Tenenbaum, C. de Melo, M. Krishna, L. Paull, F. Shkurti, and A. Torralba. Conceptfusion: Open-set multimodal 3d mapping. *arXiv*, 2023. 2

[17] V. S. Dorbala, J. F. M. J. au2, and D. Manocha. Can an embodied agent find your "cat-shaped mug"? llm-based zero-shot object navigation, 2023. 2, 7, 8

[18] O. Mees, J. Borja-Diaz, and W. Burgard. Grounding language with visual affordances over unstructured data, 2023.

[19] 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, 2022. 3

[20] Y. Xie, C. Yu, T. Zhu, J. Bai, Z. Gong, and H. Soh. Translating natural language to planning goals with large-language models. *arXiv preprint arXiv:2302.05128*, 2023.

[21] Y. Ding, X. Zhang, C. Paxton, and S. Zhang. Task and motion planning with large language models for object rearrangement, 2023.

[22] K. Lin, C. Agia, T. Migimatsu, M. Pavone, and J. Bohg. Text2motion: From natural language instructions to feasible plans, 2023. 2, 3

[23] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning (ICML)*, 2022. 2, 3, 4

[24] B. Ichter, A. Brohan, Y. Chebotar, C. Finn, K. Hausman, A. Herzog, D. Ho, J. Ibarz, A. Irpan, E. Jang, R. Julian, D. Kalashnikov, S. Levine, Y. Lu, C. Parada, K. Rao, P. Sermanet, A. T. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, M. Yan, N. Brown, M. Ahn, O. Cortes, N. Sievers, C. Tan, S. Xu, D. Reyes, J. Rettinghouse, J. Quiambao, P. Pastor, L. Luu, K.-H. Lee, Y. Kuang, S. Jesmonth, K. Jeffrey, R. J. Ruano, J. Hsu, K. Gopalakrishnan, B. David, A. Zeng, and C. K. Fu. Do as i can, not as i say: Grounding language in robotic affordances. In *Annual Conference on Robot Learning (CoRL)*, 2022. 2, 3

[25] K. Valmeekam, A. Olmo, S. Sreedharan, and S. Kambhampati. Large language models still can't plan (a benchmark for llms on planning and reasoning about change), 2023. 2

[26] W. Huang, F. Xia, D. Shah, D. Driess, A. Zeng, Y. Lu, P. Florence, I. Mordatch, S. Levine, K. Hausman, and B. Ichter. Grounded decoding: Guiding text generation with grounded models for robot control, 2023. 3

[27] Y. Jiang, A. Gupta, Z. Zhang, G. Wang, Y. Dou, Y. Chen, L. Fei-Fei, A. Anandkumar, Y. Zhu, and L. Fan. Vima: General robot manipulation with multimodal prompts. *arXiv preprint*, 2022. 3

[28] S. Huang, L. Dong, W. Wang, Y. Hao, S. Singhal, S. Ma, T. Lv, L. Cui, O. K. Mohammed, B. Patra, Q. Liu, K. Aggarwal, Z. Chi, J. Bjorck, V. Chaudhary, S. Som, X. Song, and F. Wei. Language is not all you need: Aligning perception with language models, 2023.

[29] 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. In *arXiv preprint*, 2023. 3

[30] B. Yamauchi. A frontier-based approach for autonomous exploration. In *IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, 1997. 3, 7

[31] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, and D. Zhou. Chain of thought prompting elicits reasoning in large language models. In *Neural Information Processing Systems (NeurIPS)*, 2022. 4

[32] 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 *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. 6

[33] 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. 6

[34] K. Yadav, R. Ramrakhya, S. K. Ramakrishnan, T. Gervet, J. Turner, A. Gokaslan, N. Maestre, A. X. Chang, D. Batra, M. Savva, et al. Habitat-matterport 3d semantics dataset. *arXiv preprint arXiv:2210.05633*, 2022. URL https://arxiv.org/abs/2210.05633. 6

[35] E. Wijmans, A. Kadian, A. Morcos, S. Lee, I. Essa, D. Parikh, M. Savva, and D. Batra. DD-PPO: Learning Near-Perfect PointGoal Navigators from 2.5 Billion Frames. In *International Conference on Learning Representations (ICLR)*, 2020. 7, 8

[36] R. Ramrakhya, E. Undersander, D. Batra, and A. Das. Habitat-web: Learning embodied object-search strategies from human demonstrations at scale. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 7, 8

[37] K. Yadav, A. Majumdar, R. Ramrakhya, N. Yokoyama, A. Baevski, Z. Kira, O. Maksymets, and D. Batra. Ovrl-v2: A simple state-of-art baseline for imagenav and objectnav, 2023. 7, 8

[38] R. Ramrakhya, D. Batra, E. Wijmans, and A. Das. Pirlnav: Pretraining with imitation and rl finetuning for objectnav, 2023. 7, 8

[39] D. S. Chaplot, H. Jiang, S. Gupta, and A. Gupta. Semantic curiosity for active visual learning. In *ECCV*, 2020. 7, 8

[40] B. Yu, H. Kasaei, and M. Cao. L3mvn: Leveraging large language models for visual target navigation, 2023. 7, 8

[41] A. Kadian, J. Truong, A. Gokaslan, A. Clegg, E. Wijmans, S. Lee, M. Savva, S. Chernova, and D. Batra. Sim2Real Predictivity: Does Evaluation in Simulation Predict Real-World Performance? *IEEE Robotics and Automation Letters*, 2020. 8

[42] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017. 8

[43] X. Zhou, R. Girdhar, A. Joulin, P. Krähenbühl, and I. Misra. Detecting twenty-thousand classes using image-level supervision. In *ECCV*, 2022.

[44] L. H. Li, P. Zhang, H. Zhang, J. Yang, C. Li, Y. Zhong, L. Wang, L. Yuan, L. Zhang, J.-N. Hwang, K.-W. Chang, and J. Gao. Grounded language-image pre-training. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 8