On Grounded Planning for Embodied Tasks with Language Models

Bill Yuchen Lin[1] * **Chengsong Huang** [2] * **Qian Liu**[3]
Wenda Gu[1] **Sam Sommerer**[1] **Xiang Ren**[1]
[1]University of Southern California [2]Fudan University [3]Sea AI Lab
{yuchen.lin, wendagu, sommerer, xiangren}@usc.edu
huangcs19@fudan.edu.cn, liuqian@sea.com

Abstract

Language models (LMs) are shown to have commonsense knowledge of the physical world, which is fundamental for completing tasks in everyday situations. However, it is still an open question whether LMs have the ability to generate grounded, executable plans for embodied tasks. It is very challenging because LMs do not have eyes or hands to perceive a realistic environment. In this work, we introduce the first study on this important research question. We first present a novel problem formulation named G-PlanET, which takes as input a high-level goal and a table of objects in a specific environment. The expected output is a plan consisting of step-by-step instructions for agents to execute. To enable the study of this problem, we establish an evaluation protocol and devise a dedicated metric for assessing the quality of plans. In our extensive experiments, we show that adding flattened tables for encoding environments and using an iterative decoding strategy can both improve the LMs' ability for grounded planning. Our analysis of the results also leads to interesting non-trivial findings. [2]

1 Introduction

Pre-trained language models (LMs) have shown a great ability for natural language understanding and generation tasks such as question answering, machine translation, and summarization. They indeed capture some commonsense knowledge about our physical world such as birds can fly. However, it is still an open problem whether the LMs can learn to reason in a grounded realistic environment. After all, an LM does not have eyes or hands to perceive the specific situations in real life that humans experience and interact with.

Beyond natural language processing (NLP), the area of embodied robotics learning focuses on developing AI agents that can navigate and complete tasks by interacting with real-world environments, which are often simulated by physical engines such as AI2THOR (Kolve et al., 2017). The ALFRED benchmark (Shridhar et al., 2020) is among the first datasets that connect both NLP and robotics for studying language-guided agents. They aim to develop and evaluate agents that can map language instructions to action sequences such that the agent can change the states of objects in an environment to complete a certain goal (*e.g.,* clean a thing and place it somewhere).

However, the primary focus of ALFRED and similar datasets is the understanding of pre-defined plans (*i.e.,* step-by-step instructions of a high-level goal), instead of reasoning with the realistic environment for planning by the agents themselves. Prior works are studying the ability of agents to

* The first two authors contributed equally.

[2]Our code and data will be public: https://github.com/INK-USC/G-PlanET

The 2nd Workshop on Interactive Learning for Natural Language Processing (InterNLP 2022) at 36th Conference on Neural Information Processing Systems (NeurIPS 2022).

understand and execute a plan, but not the reasoning ability to create a plan, which is a higher-level skill. Also, the role of language models is not well studied in the ALFRED benchmark and its agents, where LMs are mainly used as an encoder to embed token sequences but not for reasoning.

There are a few prior studies on the planning ability of LMs. Huang et al. (2022) show that GPT-3 and similar LMs have the ability to generate general plans for completing an everyday task. However, such general plans are not grounded in any realistic environment because the LMs are environment-agnostic. Therefore, these plans are not directly executable for agents to follow. For example, given an ALFRED task to "move a teapot from the stove to a shelf", the embodied agents expect to know where the teapot is located and how to get there. By looking at the given environment of a kitchen, humans can easily perceive that the teapot is on the stove and the current position of the agent, so a grounded plan should begin with "turn right and walk to the stove."

Can LMs also learn such grounded planning ability? How should we evaluate and improve LMs to this end? In this paper, we propose to study the ability of language models for grounded planning for embodied tasks (G-PlanET). As a step towards this, we explore a setting where the input to LMs is two-fold: a high-level task to complete and the objects of a specific environment in the form of a table. The output is a plan consisting of step-by-step low-level actions that agents will be able to directly execute. We formulate G-PlanET as a language generation problem and thus focus on studying encoder-decoder base LMs such as BART (Lewis et al., 2020).

Figure 1: The task of G-PlanET.

To create data and an evaluation protocol for G-PlanET, we re-purpose the ALFRED data by developing a suite of data conversion programs, which extract the object information from the environment and format them with tables, such that models can access observations for realistic environments. We also devise a dedicated evaluation metric named KAS that fits the problem.

As for the methods of G-PlanET, we propose to flatten an object table as a sequence of tokens and append them after the high-level goal on the input side and fine-tune the base LMs to generate plans. Plus, we propose a simple yet effective decoding strategy that iteratively generates the next step by appending the previous generation as part of the new inputs. Our experimental results and analysis show that encoding with object tables and iterative decoding are both important to improve LMs for G-PlanET. To sum up, our contributions are as follows:

- **G-PlanET as an important problem.** To the best of our knowledge, we are among the first to study LMs' ability for planning embodied tasks that are grounded in realistic environments. G-PlanET, as introduced in Sec. 2, is of vital importance for further generalizing large LMs and connecting NLP with embodied intelligence.
- **A comprehensive evaluation protocol.** We make extensive efforts to create data tables from ALFRED and AI2THOR for supporting the evaluation of G-PlanET. (Sec. 2.2) In addition, we devise a dedicated evaluation metric, KAS, for better assessing plans. (Sec. B.1)
- **Methods for improving LMs.** To improve the grounded planning ability of LMs, we present two simple yet effective components – flattening object tables and an iterative decoding strategy. Both show performance gains. (Sec. 3)

2 Problem Formulation

Here we present the problem formulation of grounded planning for embodied tasks, the background knowledge, and the data sources.

2.1 G-PlanET with LMs.

As discussed in Sec. A, the ALFRED benchmark does not explicitly test the *planning* ability, while prior works on planning with LMs have not considered *grounding* to a specific environment. In this work, we focus on evaluating and improving the ability to generate *grounded plans* for *embodied*

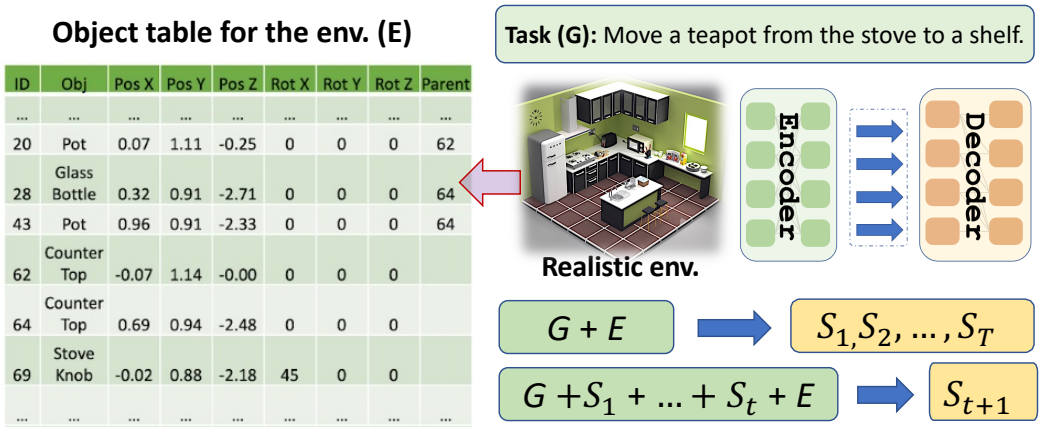

Figure 2: The overall workflow of the proposed methods. First, we extract the object table from the realistic environment. Then we flatten the table into a sequence of tokens E (Sec. 3.2). We provide two learning methods for generating plans: 1) generate the whole plan $S_1, S_2, \cdots, S_T$ and 2) iteratively decode the S_{t+1} (Sec. 3.3).

tasks with LMs, which we dub as G-PlanET. It has been an underexplored open problem for both the robotics and NLP communities.

Task formulation. The task we aim to study in this paper is essentially a language generation problem. Specifically, the input is two-fold: 1) a high-level goal G and 2) a specific environment E that the agents need to ground to. The expected output is a sequence of actionable plans $S = \{S_1, S_2, \cdots\}$ to solve the given goal in the specific environment step-by-step. The goal G and the plan S are in the form of natural language, while the environment E can be viewed as a data table consisting of the object information in a room. Figure 2 shows an illustrative example and we will discuss more details in Section 3.2.

2.2 Data for G-PlanET.

To build a large-scale dataset for studying the G-PlanET task, we re-use the goals and the plans of ALFRED and extract object information from AI2THOR for the aligned environment. The ALFRED dataset uses the AI2THOR engine to provide an interactive environment for agents with an egocentric vision to perform actions. However, the dataset does not contain explicit data about objects in the environment (*e.g.,* the coordination, rotation, and spatial relationship with each other).

We develop a suite of conversion programs for using AI2THOR to re-purpose the ALFRED benchmark for evaluating the methods shown in Section 3. We managed to get a structured data table to describe the environment of each task in the ALFRED dataset. We explore the AI2THOR engine and write conversion programs such that we can get full observations of all objects: properties (movable, openable, etc.), positions (3D coordinates & rotation), sizes, and spatial relationships (*e.g.,* object A is on the top of object B). We believe our variant of the ALFRED data will be a great resource for the community to study G-PlanET and future directions in grounded reasoning.

3 Methods

We show the task formulation of G-PlanET in Sec. 2. Here we introduce the methods that we adopt or propose to address the G-PlanET problem. First of all, we present the base language models that are encoder-decoder architectures. Then, we show in detail how we encode the environment data and integrate them with the seq2seq learning frameworks. Finally, we propose an interactive decoding strategy that significantly improves performance.

3.1 Base Language Models

Pretrained encoder-decoder language models, such as BART (Lewis et al., 2020) and T5 (Raffel et al., 2020), have achieved promising performance in many well-known language generation tasks such as

summarization and question answering. They also show great potential for general commonsense reasoning tasks such as CommonsenseQA (Talmor et al., 2019), suggesting that these large LMs have common sense to some extent. As the G-PlanET can be also viewed as a text generation problem, we use these LMs as the backbone for developing further planning methods, hoping that their common sense can be grounded in real-world situations for embodied tasks.

Vanilla baseline methods. The simplest and most straightforward baseline method of using such LMs to solve G-PlanET is to ignore the environment and only use the goal as the sole input. Then, we fine-tune the base LMs with the training data and expect they can directly output the whole plan as a single sequence of tokens (including special separator tokens).

3.2 Encoding Realistic Environments

To enable the LMs to perceive an environment, we need to encode the object tables described in Sec. 2.1. Following prior works in table-based NLP tasks (Chen et al., 2020; Liu et al., 2022b), we flatten a table into token sequences row by row, thus creating a linearized version of an object table. Then, we append the flattened table after the goal to form a complete input sequence. Thus, the input side of the encoder-decoder finally has the environment information for generating a grounded plan.

Considering the max sequence limit, we only choose to encode objects by their **type**, **position**, **rotation**, and the **receptacle** parent. The object type does not only tell what an object is but also implies commonsense affordance (*e.g.,* a microwave can heat up something, a knife can slice something) which is very important for planning. The position information is essential for agents to navigate and find objects, thus playing an important part in planning. The rotation is also useful for some objects that can only be used with a certain orientation (*e.g.,* a refrigerator can only be opened when the agent is in front of it). The receptacle of an object and itself has a close spatial connection (*e.g.,* a pen is on a desk; an apple is in a fridge). Every object has a unique identifier such that objects of the same type can be referred to precisely when they are receptacles of others. In addition, the agent is represented as a special object.

3.3 Iterative Decoding Strategy

Adding the flattened table of object information to the input sequences indeed improves the LMs in terms of their perception of the realistic environments, which forms the foundation of grounded planning. However, the thinking process is still limited by the conventional seq2seq learning framework, which assumes the LMs should output a complete plan by a single pass of decoding. We argue that a thoughtful planning process should carefully handle the coherence of each step, otherwise errors will accumulate and cause a failed plan.

Therefore, we propose a simple yet effective decoding strategy that learns to iteratively generate a plan step by step. Specifically, we append previously generated steps until the current step t to the input sequence (*i.e.,* `Input` = $[G + S_1 + \cdots + S_t(+E)]$) for generating the next step (*i.e.,* `Output` = S_{t+1}). This iterative decoding process will end until the LM generates the special token `END`. In the training stage, we use the ground-truth references for $S_{\leq t}$; in the inference stage, we do not have such references, so we use the model predictions as $S_{\leq t}$.

Notably, in contrast to the conventional seq2seq learning process, the iterative decoding strategy needs to run the encoder-decoder model $N + 1$ times to generate a plan with N steps. The additional computation cost for re-encoding is worthy. Imagine when we humans are planning a task in a room. It is natural for us to come up with the plans step by step, and it is very likely that the most useful information to generate different steps is about different objects. Therefore, a temporally dynamic attention mechanism is favorable in planning with LMs. Our iterative decoding strategy encourages the encoder-decoder architectures to learn such ability.

3.4 Other Methods

Pretrained table encoders. Since we use environmental information in a tabular format and BART has not been pre-trained in the tabular form of input, BART may not be able to use this part of information well. Therefore, we employ TAPEX (Liu et al., 2022b), the state-of-the-art pre-trained language model on tabular data. Using SQL execution as the only pre-training task, TAPEX achieves

Data Split →	Unseen Room Layouts			Seen Room Layouts		
Methods ↓ Metrics →	CIDEr	SPICE	KAS	CIDEr	SPICE	KAS
BART-base (vanilla)	0.9417	0.1378	0.2455	0.8231	0.1277	0.2197
BART-large (vanilla)	1.4632	0.3168	0.4069	1.4414	0.3161	0.3900
GPT-J-6B	1.1968	0.2655	0.3622	1.1047	0.2509	0.3370
BART-base w/table	1.6706	0.3692	0.4584	1.6230	0.3595	0.4339
BART-large w/table	1.6630	0.3491	0.4411	1.5865	0.3393	0.4204
BART-large (TAPEX)	2.8824	0.5054	0.6373	2.7432	0.4944	0.6045
BART-base w/table + iterative decoding	**2.9147**	0.5107	0.6334	2.8582	**0.5118**	0.6124
BART-large w/table + iterative decoding	2.8580	0.5194	**0.6518**	**2.8799**	0.5096	**0.6326**
BART-large (TAPEX) + iterative decoding	2.8440	**0.5210**	0.6313	2.6959	0.5036	0.6074

Table 1: Experimental results for the G-PlanET by different base LMs. The methods are grouped by model types and whether encoding the environment; by decoding strategies.

better tabular reasoning capability than BART, and thus we expect TAPEX can make full use of the environmental information represented by the table in our task.

In-context few-shot learning with GPT-J. Finally, to explore whether large-scale language models can master the task with few-shot examples, we also experimented with few-shot performance on a larger language model GPT-J 6B (Wang and Komatsuzaki, 2021).

4 Evaluation

How do we evaluate a method for G-PlanET? Due to the novelty of the problem, it is challenging to evaluate and analyze its methods. We present a general evaluation protocol and a complementary metric to measure the quality of generated plans. We report the main experimental results with the proposed evaluation protocol. In Appendix, we show in detail the evaluation metrics (Sec. B.1), experimental setups (Sec. B.2), and implementation details. We leave the analysis in Sec. C.

We report the main results in Table 1, and leave the deeper analysis in the next section. To sum up, we find that encoding the object table as part of the inputs will significantly improve the performance, and pre-training on other table-related tasks can benefit G-PlanET a lot. The iterative decoding strategy is also an important component that can further improve the results to some extent.

Effect of model sizes. Table 1 shows that small models can perform as well or even better than large models in some cases. This is mainly due to the following reasons. 1) The sentences in plans are relatively simpler than other NLG tasks, with a smaller vocabulary and shorter length. This leaves the power of large models in terms of generation unexpressed, 2) G-PlanET is a task to examine the ability to plan rather than write. Whether this ability changes with model size remains to be explored. 3) For scenarios with the table, the form of the task is not the same as the traditional generation task, so the training phase will have a greater impact. Models with fewer parameters are more sufficiently tuned with limited data.

5 Conclusion

In this paper, we present the very first study on grounded planning for embodied tasks with language models. The G-PlanET problem is of great importance for the development of embodied intelligence of LMs, which we believe is fundamental for artificial general intelligence. To investigate the G-PlanET ability of encoder-decoder LMs, we create a comprehensive evaluation protocol as well as a dedicated metric, KAS for assessing the quality of generated plans. Moreover, we present two methods for improving the LMs' ability for G-PlanET — flattening object tables and iterative decoding. We perform extensive experiments and analyses for verifying their effectiveness and obtaining non-trivial findings. We believe that our paper will spur further research on studying G-PlanET and continual exploration for connecting LMs and embodied tasks in realistic environments.

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. 2022. Do as i can and not as i say: Grounding language in robotic affordances. volume abs/2204.01691.

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2016. Spice: Semantic propositional image caption evaluation. In *ECCV*.

Sercan Ö. Arik and Tomas Pfister. 2021. Tabnet: Attentive interpretable tabular learning. *ArXiv*, abs/1908.07442.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, P. Abbeel, A. Srinivas, and Igor Mordatch. 2021. Decision transformer: Reinforcement learning via sequence modeling. In *NeurIPS*.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2020. Tabfact: A large-scale dataset for table-based fact verification. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Wael A. Farag and Zakaria Saleh. 2018. Behavior cloning for autonomous driving using convolutional neural networks. *2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, pages 1–7.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. TaPas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333, Online. Association for Computational Linguistics.

Wenlong Huang, P. Abbeel, Deepak Pathak, and Igor Mordatch. 2022. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *ICML*.

Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. 2017. AI2-THOR: An Interactive 3D Environment for Visual AI. *arXiv*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Bill Yuchen Lin, Wangchunshu Zhou, Ming Shen, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. 2020. CommonGen: A constrained text generation challenge for generative commonsense reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1823–1840, Online. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Hao Liu, Yang Liu, Hong He, and Hang Yang. 2022a. Lebp - language expectation & binding policy: A two-stream framework for embodied vision-and-language interaction task learning agents. *ArXiv*, abs/2203.04637.

Qian Liu, Bei Chen, Jiaqi Guo, Morteza Ziyadi, Zeqi Lin, Weizhu Chen, and Jian-Guang Lou. 2022b. TAPEX: Table pre-training via learning a neural SQL executor. In *International Conference on Learning Representations*.

So Yeon Min, Devendra Singh Chaplot, Pradeep Ravikumar, Yonatan Bisk, and Ruslan Salakhutdinov. 2021. Film: Following instructions in language with modular methods. *ArXiv*, abs/2110.07342.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Pardis Pashakhanloo, Aaditya Naik, Yuepeng Wang, Hanjun Dai, Petros Maniatis, and Mayur Naik. 2022. Codetrek: Flexible modeling of code using an extensible relational representation. In *International Conference on Learning Representations*.

Alexander Pashevich, Cordelia Schmid, and Chen Sun. 2021. Episodic transformer for vision-and-language navigation. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15922–15932.

Xinyu Pi, Qian Liu, Bei Chen, Morteza Ziyadi, Zeqi Lin, Yan Gao, Qiang Fu, Jian-Guang Lou, and Weizhu Chen. 2022. Reasoning like program executors. *ArXiv preprint*, abs/2201.11473.

Colin Raffel, Noam M. Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *ArXiv*, abs/1910.10683.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2020. ALFRED: A benchmark for interpreting grounded instructions for everyday tasks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 10737–10746. IEEE.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew J. Hausknecht. 2021. Alfworld: Aligning text and embodied environments for interactive learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 4566–4575. IEEE Computer Society.

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. 2022. Scienceworld: Is your agent smarter than a 5th grader?

Xin Wang, Jiawei Wu, Junkun Chen, Lei Li, Yuan-Fang Wang, and William Yang Wang. 2019. Vatex: A large-scale, high-quality multilingual dataset for video-and-language research. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 4580–4590. IEEE.

Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Josh Tenenbaum. 2018. Neural-symbolic VQA: disentangling reasoning from vision and language understanding. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 1039–1050.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. TaBERT: Pretraining for joint understanding of textual and tabular data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8413–8426, Online. Association for Computational Linguistics.

Ori Yoran, Alon Talmor, and Jonathan Berant. 2022. Turning tables: Generating examples from semi-structured tables for endowing language models with reasoning skills. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6016–6031, Dublin, Ireland. Association for Computational Linguistics.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with BERT. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

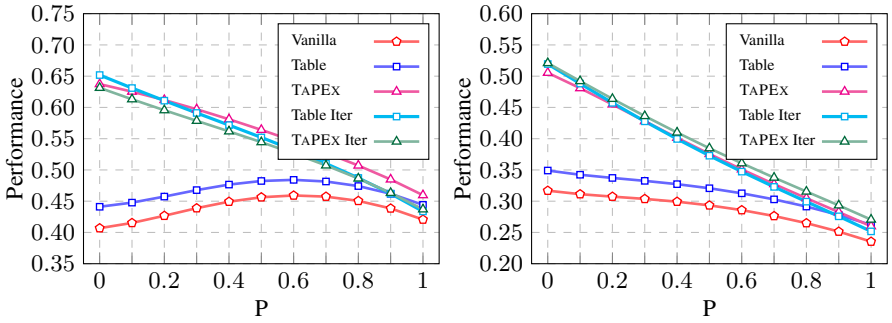

Figure 3: The step-wise reweighting results of KAS (**Top**) and SPICE (**Bottom**).The x-axis indicates the parameter p in the geometric distribution and also the importance of the preceding step, and the y-axis indicates the weighted result of each step. A larger coefficient means that the previous step is more important.

A Background Knowledge

Embodied tasks. The ALFRED benchmark (Shridhar et al., 2020) is among the first benchmarks focusing on embodied tasks in realistic environments, although most of the examples are *household* tasks. It aims to test the ability of agents to execute embodied tasks in real-world scenarios. Specifically, the agents need to understand language-based instructions and output a sequence of actions to interact with an engine named AI2-THOR (Kolve et al., 2017), such that the given tasks can be achieved.

Language instructions. Language instructions play an important role in the ALFRED benchmark. The embodied tasks are annotated with a high-level goal and a low-level plan (*i.e.,* a sequence of executable actions for robots) in natural language, which are both inputs to the agents. The agents need to understand such language instructions and parse them into action templates. Note that the agents do not need to *plan* for the task, as they already have the step-by-step instructions to follow.

Task planning. Prior works show that large pre-trained language models (LMs) such as GPT-3 (Brown et al., 2020) can generate general procedures for completing a task. However, such plans are not aligned with the particular environment in which we are interested. This is because these methods never encode the environment as part of the inputs to LMs for grounding the plans to the given environment. Therefore, such non-grounded plans are hardly useful in guiding agents to work in real-world situations.

B Evaluation (cont.)

B.1 Metrics

Step-wise evaluation. Conventional evaluation metrics such as BLEU (Papineni et al., 2002) and ROUGE (Lin, 2004) measure the similarity between generated text and truth references as a whole, which is suitable for translation and summarization. However, the output text of planning tasks such as our G-PlanET is highly structured. A plan naturally can be split into a sequence of step-by-step actions. Using the conventional way to evaluate plans inevitably breaks such internal structures and will lead to inaccurate measurement. For example, if the first step of the generated plan is the same as the last step of the reference plan, the conventional evaluation will still assign a high score to such a generated plan, even though it is not useful at all. Therefore, we argue that it is much more reasonable to evaluate the similarity of a pair of plans step by step. Specifically, we first align the generations and the truths and compute the scores of every step[3] by multiple metrics. Then, we aggregate the

[3]The ALFRED authors ensure that the references consist of atomic action steps and all references share the same length. Therefore, we consider the length of truth plans as the standard: when the generated plan has more steps than the truth plans, we cut off them; when the generation has fewer steps than the references, we duplicate the last step to make them even for step-wise evaluation.

final score by taking the average of all steps. We also consider other temporal weighting aggregation for more analysis in Sec. C.

Measuring grounded plans. It is a unique challenge for evaluating G-PlanET to consider the grounding nature of plans. Metrics, such as BLEU, METEOR, and ROUGE, do not give a suitable penalty when a plan is similar to the reference in terms of word usage, yet leading to totally different states in an interactive environment for embodied tasks. For example, it is only a one-word difference between "turn to the *left*" vs "turn to the *right*", but the agents that faithfully follow these instructions can arrive at very different places.

The LM-based metrics, *e.g.,* BERTScore (Zhang et al., 2020), are not suitable either because the neural embeddings of "left" and "right" are also very similar. Plus, the grounded plans for G-PlanET are object-centric in a context and very similar to the captions of a sequence of events by visual perception, for which these metrics are not specifically designed. Considering these limitations, we use two typical metrics that are widely used for captions and devise a new metric for complementary measurement.

The first two metrics are **CIDEr** (Vedantam et al., 2015) and **SPICE** (Anderson et al., 2016), which are both widely used for tasks where the outputs are highly contextualized and describe natural scenarios in everyday life, *e.g.,* VaTex (Wang et al., 2019) and CommonGen (Lin et al., 2020). In particular, SPICE parses both the generation and references to scene graphs, a graph-based semantic representation. Then, it calculates the edge-based F1 score to measure the similarity between each step. Note that SPICE computation has a special focus on the propositions. This is particularly favorable for evaluating G-PlanET since there are many actions in the grounded plans, where propositions can be seen as atomic units for evaluation.

KeyActionScore (KAS). Inspired by SPICE, a step in a plan can be deconstructed into several propositions that are represented as edges. However, not all propositions in SPICE are necessarily important in evaluating plans for G-PlanET. Not to mention that SPICE relies on an external parser that is expensive to run yet sometimes contains noisy outputs. Also, most of the truth plans in the ALFRED annotations are overly specific, and it is not necessary for a plan to cover all details. Therefore, we devise a metric that focuses on the key actions of the generated plans and checks if they are part of references, named *Key Action Score* (**KAS**).

Specifically, we extract a set of key action phrases from each step in the generated plan $\hat{S}_i$ and the truth reference S_i respectively. We denote this two sets as $\hat{S}_i = \{\hat{a}_1, \hat{a}_2, \cdots\}$ and $S_i = \{a_1, a_2, \cdots\}$. Then, we check how many action phrases in $\hat{S}_i$ are covered by the truth set S_i, the precision then becomes the KAS score for the i-th step in the plan. To increase the matching quality, we curate a set of rules and a dictionary to map the actions that share the same behaviors. For example, "*turn* to the *left*" and "turn left" are counted as a single match; "*go* straight" and "*walk* straight" can be matched too. In addition, we break the compound nouns such that we allow partial scores to match for a smoother scoring (*e.g.,* "xxx on the *table*" vs "xxx on the *coffee table*"). Simply put, the KAS metric looks at the key actions extracted from the plans and checks if these important elements can be (fuzzy) matched to count as a valid step.

B.2 Experimental Setup

Data statistics. Table 2 shows some statistics of our dataset that we described in Sec. 2.1. We follow the data split in ALFRED to split the train, valid, and test dataset. The data split is based on whether the room layout has been seen in the training tasks. It is usually easier for robotic agents to map instructions to low-level actions in seen rooms than in unseen rooms. However, for the planning ability that we want to study with G-PlanET in this paper, the two splits do not differ very much. We keep using this split to make the results consistent and convenient for people who want to connect our results with the ALFRED results.

Implementation details. In single-pass decoding, we format the output sequences as follows: "`Step 1:`$[S_1]$ | `Step 2:` $[S_2]$ | $\cdots$ | `END`". When appending the flattened table of objects, we format input with "$[G]$ `Env:` `[row 1]` `[SEP]` `[row 2]` $\cdots$", where the `[row i]` is a sequence of the i-th object including its id, type, coordinates, rotation, parent receptacles, etc.

split →	train	valid		test	
aspect ↓	-	seen	unseen	seen	unseen
# tasks	21,025	820	821	705	694
avg. $\|G\|$	9.26	9.32	9.26	10.3	9.95
avg. # O	73.71	74.21	77.91	75.31	73.9
avg. # T	6.72	6.79	6.26	6.95	6.63
avg. $\|S_i\|$	11.24	11.13	11.49	9.84	10.19

Table 2: The avg. $|G|$ means the average length of goal and the avg. $|S_i|$ means the average length of each step. The avg. # O is the average number of objects in each room and the avg. # T is the average number of steps.

Hyper-parameters. For BART, we conducted experiments with both BART-base and BART-large to explore the effect of the size of the language model on the planning ability. The models are with a batch size of 4, a learning rate of 3×10^{-5}, and the AdamW optimizer (epsilon 1e-8). The number of epochs is 5 for no-iterative models and 3 for iterative ones because of the bigger training set. The fine-tuning of all models based on TAPEX lasts up to $20,000$ steps with a batch size of 24. We follow the default learning rate scheduling strategy with a peak learning rate of 3×10^{-5}. For the GPT-J model, we randomly selected 5 training examples and used them as prefixes for each test example. The model is allowed to output at most 250 tokens.

Computational cost. ALL experiments about BART are performed on RTX 6000 GPU cards. The iterative generation model may cost about 24 hours to train on BART-large. The other experiment will cost less than 10h. All experiments about TAPEX and GPT-J are performed on a Tesla V100 or A100 GPU card with a training time less than 12 hours, and all pre-trained language model weights are downloaded from the Huggingface.

C Analysis

In this section, we deeply analyze the performance of the methods in Table 1 from multiple aspects and provide non-trivial findings that can help future research. For a fair comparison, all analytical experiments were performed in the BART-large model on the unseen split of the test data.

C.1 Temporal Re-weighting of Scores

When we computed the overall score of a plan with a metric, we use the average score to aggregate the score for each step. However, in a realistic environment, there are causality constraints for an agent to complete the steps – *i.e.,* some tasks can only be done when their prerequisite steps are finished. For example, only when the agent arrives at the microwave can it heat the bread in its hands.

Therefore, the earlier steps in a plan should be of higher importance, while our previous evaluation is based on a uniform distribution of the weights across steps. To this end, we adopt *geometric distribution* to re-weight the step-wise importance for weighted aggregation. The geometric distribution can be used to model the number of failures before the first success in repeated mutually independent Bernoulli trials, each with a probability of success p.

$$f(x) = p(1-p)^x \qquad (0 < p < 1)$$

This suits our setup well because when the first step is incorrect, the whole task can hardly be completed and executed in a generated plan for ALFRED. The range of p in the original setting of the geometric distribution is restricted to between 0 and 1. When $p = 0$, each step has the same weight (uniform importance), which is exactly what we have done in Tab. 1. When $p = 1$, the first step is the only thing we look at for evaluation, meaning that the other steps will be given zero weights for aggregation.

Figure 3 shows the results on unseen subset which is more realistic. The performance of the iterative and non-iterative approaches is very close in the case of the first step. This is mainly because iterative methods are similar to non-iterative methods when generating the first step, and differ only after the second step. At the same time, it can be seen that there is an overall downward trend in performance

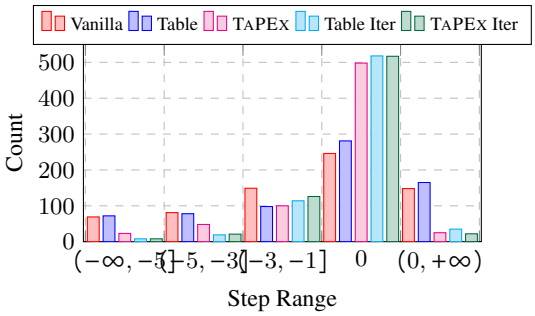

Figure 4: The result of error statistics for # of step.

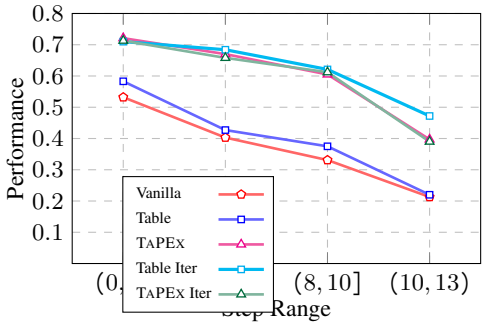

Figure 5: The result of KAS of tasks with a different number of steps. Due to the large variance caused by the small number of samples of certain lengths, we use the statistics by dividing the intervals.

as the focus moves to the early step. The main reason is that the later the subtask is, the closer it is to the high-level instruction. For example, if the task goal is to place the sponge in the sink, the final step must be to place the sponge in the sink. This feature makes the last step of subtask generation very simple, resulting in high performance. We also see that the performance of the non-iterative method rises and then falls in KAS, and the change in a downward trend in SPICE. The main reason is an error in the number of steps in the non-iterative method, which will be explained next.

C.2 Error Analysis on the Lengths of Plans

In our experiments, we found a huge gap in the prediction of the number of task steps between iterative and non-iterative methods, which may be an important reason for the final performance difference. As shown in Figure 4, iterative methods have a higher probability of predicting the number of steps for the correct task, while non-iterative methods do underestimate the number of steps. In our evaluation framework, the missing follow-up steps of non-iterative methods are often generated by copying. This is one of the reasons for the poor performance of non-iterative methods and the reason why the performance of non-iterative methods increases first in the reweight step process.

C.3 Impact of Task Length on Performance

Although all the tasks in the dataset are part of daily life tasks, they differ in difficulty. A simple metric to evaluate the difficulty of a task is the number of steps they require. Figure 5 illustrates the decrease in the quality of the generated steps as the number of task steps increases. The figure also reflects the relatively small difference in the performance of the different methods on shorter tasks. And the performance of all methods degrades rapidly on the longest tasks. The iterative approach has more significant performance benefits on longer tasks. This may be because this approach makes better use of the state changes due to intermediate steps and fixes some previous errors.

D Related Work

Grounded commonsense reasoning. ALFWorld (Shridhar et al., 2021) also uses LM to generate the next step in a text game which is based on ALFRED. SciWorld (Wang et al., 2022) designed a text game to find whether the LMs have learned to reason about commonsense. SayCan (Ahn et al., 2022) also uses LM to find the potential next step in the real world. Both these three works only expect to learn the next step in a text game. Our methods share similar motivation with decision transformer (Chen et al., 2021) and Behavior Cloning (Farag and Saleh, 2018), but we work on very different applications.

Table-based NLP. Our work is closely related to two lines of tabular data usage in NLP: the approach to modeling tabular representations and the application of a table as an intermediate representation. For the first line of work, there is rich literature focusing on modeling tabular representations, including TabNet (Arik and Pfister, 2021), TAPAS (Herzig et al., 2020), TaBERT (Yin et al., 2020) and TAPEX (Liu et al., 2022b). We have explored the impact of state-of-the-art table representation models (e.g., TAPEX) on our task in experiments. As for the second line of work, previous work has explored to use of tables in several downstream tasks, including visual question answering (Yi et al., 2018), code modeling (Pashakhanloo et al., 2022), and numerical reasoning (Pi et al., 2022; Yoran et al., 2022). Different from them, our work is the first to explore the use of tabular representations in embodied tasks.

ALFRED Agents. Some previous research has been published on embodied tasks in realistic environments since the appearance of ALFRED. E.T. (Pashevich et al., 2021) first encoded the history with a transformer to solve compositional tasks and proved that pretraining and joint training with synthetic instructions can improve performance. FILM (Min et al., 2021) proposed an explicit spatial memory and a semantic search policy to provide a more effective representation for state tracking and guidance. LEBP (Liu et al., 2022a), the currently published SOTA method, generated a sequence of sub-steps by understanding the language instruction and used the predefined actual actions template to complete the sub-steps. We also try to use these methods to evaluate our generated low-level instructions. However, due to the limited importance of the low-level instructions, there is no gap with conspicuousness between our generated instructions and the ones in ALFRED.

E Limitations

The main limitations of this work on the new task G-PlanET are as follows:

- **Evaluation**: Although we have adopted and devised automatic metrics for evaluating methods for G-PlanET, there is not yet a straightforward way for us to test the ultimate success rates of such plans (if they are executed by oracle agents). We have tried to use state-of-the-art ALFRED agents such as FILM Min et al. (2021), but they did not show obvious differences using such step-by-step instructions. We believe more human evaluation will help us further refine the metrics, which can be very expensive though.
- **Methods:** Flattening object tables into sequences of tokens row by row is straightforward but might not be optimal. The number of objects can be huge for a complicated room. How can we narrow down the important objects at each step? We argue that a more advanced version of attention modules for dynamic table encoding is needed. We may not need to input the whole table for decoding at all steps. As a preliminary study, we created a retrieval augmentation method that only includes the oracle objects (that are mentioned in the next step) as the input, but we see little improvement. We think more physical rules and math computation with the object features will help us gain more improvement.