

Towards Natural Language Reasoning for Unified Robotics Description Format Files

Stanford CS224N Custom Project

Neil Nie, Austin Patel, Aakash Mishra
{neilnie, auspatel, aamishra} @stanford.edu
Stanford University

Abstract

We develop a language model that is capable of semantic reasoning with Universal Robot Description Format (URDF) files. These files describe the geometry and articulation of 3D objects. Leveraging the Sapien dataset of household items, we develop a model that can interpret URDF files alongside natural language queries to answer questions about object articulation, dynamics, and appearance. By parsing URDF files and understanding their semantic information, our model can facilitate applications like similarity search, component retrieval, and object question and answering. Our approach involves generating image renders of URDF objects, obtaining natural language captions via VLM, and then creating question-answer pairs for model training. We experiment with finetuning pre-trained large language models with different datasets while also exploring the synergy between textual and geometric data representations to enhance question-answering capabilities. We also demonstrate quantitative and qualitative results showcasing the improvement of our fine-tuned model on both a general Q&A dataset as well as a joint counting task which highlights the model's ability to reason with URDFs.

1 Key Information to include

- Mentor: Yuhui Zhang
- External Collaborators (if you have any): None
- Sharing project: No
- Contributions
 - Neil: Built the simulation environment in PyBullet. Generated the image data using the SAPIEN dataset. Generated the joint counting task training data. Ran evaluations on the joint counting task. Contributed to the writing process.
 - Austin: Setup the dataloading and training pipeline. Fine-tuned the GPT2 models (several varieties) on 2 tasks (joint counting and general QA). Contributed to the writing process.
 - Aakash: Generated the general QA task training data using GPT4-Vision. Ran the BLEU evaluation, embedding similarity, and WUPS evaluation. Contributed to the writing process.

2 Introduction

Motivation. Universal Robot Description Format (URDF) files are utilized widely to describe the geometry and articulation of 3D objects, particularly in the realm of robotics and simulation Johannessen et al. (2019). However, the potential for semantic reasoning with these descriptions and integrating natural language processing (NLP) for enhanced human-computer interaction remains an open space. In a survey taken of robotics professionals regarding working with URDF files, a highlighted concern was the difficulty in understanding and debugging file information due to the nature of its syntax-heavy design Tola and Corke (2023). In this work, we bridge this gap by

introducing a novel language model specifically tuned to understand and reason about URDF files through natural language queries. (Note, that the term "articulation" is used to refer to the physical motion properties of URDF objects.)

Object Semantics vs. Syntax. Existing work in semantic object understanding often relies on images or language-image pairs. This approach has a few limitations, namely that images are difficult to generate, and visual input might not fully capture the object articulation and geometry due to partial observability Saba (2023). On the other hand, URDF files provide rich semantic content about the object by capturing syntax-heavy information regarding each part and joint.

Data. We believe that we can extend the capabilities of code understanding with NLP to syntax-heavy files, such as URDFs. This would allow the model to not only take in natural language input in the form of labels but also precise XML syntax that describes the objects. Leveraging the "household items" Sapien dataset, our model parses URDF files to interpret their geometric and dynamic properties, facilitating a range of applications such as component retrieval and question-answering. We finetune pre-trained large language models (LLMs) with Q&A data that is generated by using GPT-4 vision using an image rendering of the URDF and prompt tuning for questions regarding articulation. We give the question-answer pairs about object dynamics and syntax from URDF XML files as finetuning input to our model to improve its performance in answering complex queries about objects.

Contribution. Our experiments demonstrate the effectiveness of our approach, with the fine-tuned model showcasing the ability to reason through both captions and URDF file contents. The quantitative results highlight significant improvements in answering questions related to object articulation dynamics and appearance, underscoring the model’s reasoning capabilities for URDF files.

3 Related Work

Language Models for Code Understanding Recent advancements in large language models have significantly enhanced their capabilities in understanding and generating code. The work of Radford et al. Radford et al. (2019a) introduced GPT-2, a foundational model that showcased the potential of unsupervised learning in generating human-like text, laying the groundwork for applying LLMs to interpret and process coding languages and structures. Furthermore, models like CodeGen by Li et al. Li et al. (2022) and PanGu-Coder by Christopoulou et al. Christopoulou et al. (2022) have been specifically fine-tuned to understand and generate programming code, demonstrating the growing intersection of natural language processing (NLP) and software engineering. This project extends these efforts by exploring the use of LLMs to parse and reason about URDF files, a specialized XML format used to describe the geometry and dynamics of objects used in robotics. Approaches like WebFormer or Lomshakov et al. (2023) have special encoding for XML tags and their parent/child relationships for use in transformer networks which help extract semantic details from syntax-rich XML files.

Vision-Language Models for Object Recognition and Captioning The integration of vision and language has led to the development of multi-modal models capable of understanding objects and their attributes through natural language. One foundational work in this area ViLBERT by Lu et al. Lu et al. (2019), is a model that processes visual and textual inputs in parallel through co-attentional transformer layers, demonstrating significant improvements in tasks like visual question answering and object recognition. Concurrently, Tan and Bansal Tan and Bansal (2019) developed LXMERT, which further refines the understanding of object-centric scenes through cross-modal pre-training across a range of vision and language tasks. More recently, the OSCAR model by Li et al. Li et al. (2020) has shown how object-semantics can enhance language and vision tasks by leveraging object tags as anchor points between images and text, leading to superior performance in image captioning and visual question answering. These works highlight the capability of VLMs to not only perceive and segment visual content but also to infer and reason about the interplay of objects within scenes. However, nearly all of these works rely on visual input and cannot directly reason about object articulation and semantics on some standardized input format, such as URDFs.

GPT4-Vision, as mentioned in the project, represents a leap in vision-language models, capable of generating detailed descriptions and question-answer pairs based on visual inputs. In our method,

we leverage this functionality to auto-generate captions and question/answer pairs on the SAPIEN dataset by Xiang et al. (2020), which offers a rich collection of 3D objects for simulation and interaction studies. The project utilizes visual data with textual descriptions by leveraging a pre-trained VLM, enabling a more nuanced understanding of object dynamics and articulations to create our Q&A dataset.

Semantic Reasoning and Object Articulation in Robotics Understanding the geometry and dynamics of 3D objects is crucial for various applications in robotics and AI. The project’s focus on semantic reasoning with URDF files for object articulation dynamics is similar to efforts to enhance robot interaction and simulation environments, as evidenced by the development of the SAPIEN engine Xiang et al. (2020). Unlike prior works, our method of automatically generating question-answer pairs as training data, and then directly finetuning large language models to reason about the physical properties and possible actions of objects through LLMs and VLMs marks a significant step toward more intelligent and versatile robotic systems.

4 Approach

Architecture (i). We used the 124M GPT2 model (Radford et al., 2019a) without architecture modification and input a concatenation of the URDF XML, VLM caption, and questions as the input tokens into our decoder which outputs the answer to the question we give. The model architecture and an example of this task are shown in Figure 1.

Given the 1024 token context window of GPT2 and the long length of many URDF files in our dataset, we opt for a URDF pre-processing step that extracts "joint-tag" information from the files and discards other tags. Specifically, for each joint, we extract the following attributes: name, joint type (prismatic, continuous or revolute), axis, and connected parent/child links. In doing so, we can process a majority of the existing input sequences within our limited context window. For sequences that exceed the token limit, we truncate the input starting with the URDF tokens.

Question-Answer Generation (ii). For the model to reason about object dynamics and semantics given a URDF XML, we need to train the model on question-answer pairs for objects in the training set. These QA pairs should reveal to the model important object semantics, such as articulation, geometry, and use-case. Example:

Question: Can this object be extended more than 30 degrees to cut paper?
 Input: Scissor URDF (XML) and label - "a pair of aluminum scissors"
 Output: Yes, it can.

We built an annotation system that queries a VLM model (GPT4-Vision) by asking it to generate a series of question-answer pairs about the image of the rendered URDF. First, we load the URDF of the object in the PyBullet simulator. We place four virtual cameras around the object to capture a complete view of the scene. We combine the four images into one and pass that into the VLM prompt. We then parse the output of the VLM into structured data and store the generated QA pairs along with the URDF XML. We were able to create around 1000 image samples given the Sapien dataset and around 3000 object-question pairs.

We additionally construct a second dataset specifically for a joint counting task in which the model is required to output the number of joints in the URDF file. The purpose of this is to demonstrate the models’ ability to reason about the contents of the URDF file which is passed as input. We generate this dataset by counting the number of joints in each URDF and using a templated question-answer format as follows:

Question: How many joints does this object have?
 Input: Cabinet URDF (XML) and label - "a wooden cabinet with multiple drawers"
 Output: This object has four joints.

Finetuning (iii). We start with pre-trained weights for GPT2-124M (Radford et al., 2019b), and GPT2-774M and finetune these weights on both the general question-answering task and the joint counting task to create two separate finetuned models for each task.

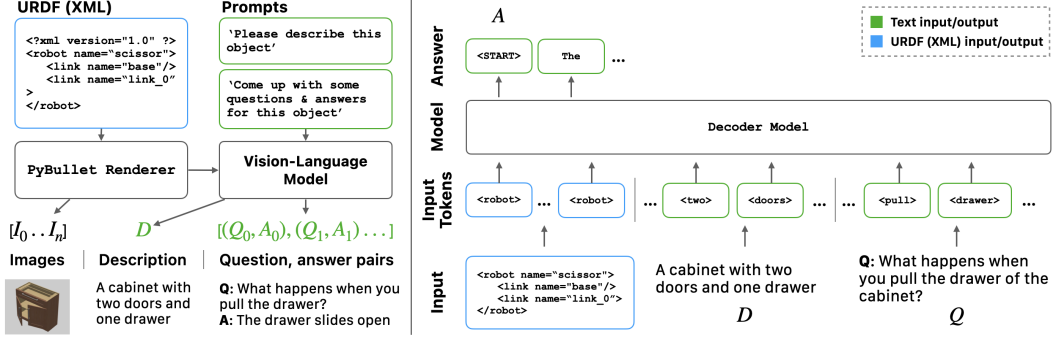


Figure 1: The architecture diagram. (left) The data labeling process with the simulator and an open-source vision-language model. (right) The large language model finetuning process, supervised by auto-generated question and answer pairs, is conditioned on a specific input URDF file.

Baseline (iv). We compare the fine-tuned version of our GPT-2 variant versus the non-finetuned versions of each of the language models we use as our pre-training base. In the following experiments section, we demonstrate the benefits of this finetuning stage.

5 Experiments

Data. We used the Sapien Xiang et al. (2020) dataset, which contains 2,346 URDF files across 46 categories mainly consisting of household objects (ex: bottle, box, chair, clock, door, etc.) as well as kitchen appliances (oven, coffee machine, dishwasher, etc.). We also queried a pre-trained visual-language model (VLM) (GPT4-Vision) to provide a natural language caption for each of these URDFs given the rendered image. Additionally, we also used the VLM to generate question-answer pairs regarding the specific object. Thus our dataset consists of tuples formatted as (URDF files, captions from VLM, questions from VLM, answers from VLM), to supervise our model.

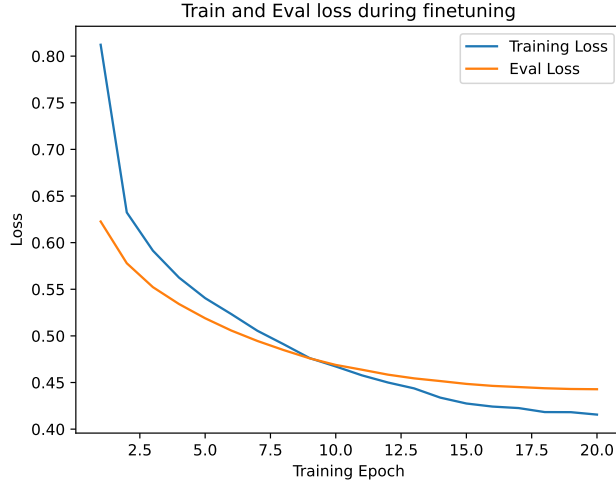


Figure 2: **Loss during training.** We train the model for 20 epochs and observe near convergence in both the training and evaluation splits.

Evaluation Method. We evaluate the decoder output by using the BLEU score Papineni et al. (2002) between the GPT-4-vision provided answer label as the reference text and the decoder output as the evaluation text. We use n-gram weights of 0.3 and 0.7 for 1-gram and 2-grams respectively and calculate the score over each data split, as well as by object category. We also use the Wu-Palmer Similarity Score (WUPS) and Cosine similarity with GPT-3 embeddings of size (53) to evaluate the model since we foresaw the drawback with only using one referential answer as a scoring mechanism. Using similarity metrics helps us attain a more granular evaluation of the model’s correctness.

The first evaluation task is the general QA task, as described in the Approach section. The model is given a question and XML object and is asked to reason about the physical properties of the object, such as the articulation of the joints or how the object can be used.

The second task joint counting, is designed to evaluate our model’s ability to understand and reason about the structural aspects of URDF-described objects. This task specifically focuses on assessing the model’s capacity to accurately identify and count the number of movable joints within a given object (some are tagged with a "fixed" attribute indicating they do not move), as opposed to merely tallying the occurrences of the term "joint" in the URDF XML files. This distinction is crucial because movable joints are key to understanding an object’s potential articulations and mechanical functions, which are fundamental for applications in robotics and automated systems.

The joint counting task aims to evaluate the model’s semantic understanding of URDF files. Accurately identifying these elements requires the model to demonstrate its retrieval capabilities.

Experimental Details. The GPT2 model and pre-trained weights are from Hugging Face and further finetuned using the Transformers pipeline library. We trained the variants of the GPT2 models on 4 A100 GPUs using data distributed parallel (DDP). As we scale up our models, using DDP is essential to the training process. We fine-tuned the model with a batch size of 8 examples per GPU, and total epochs of 20, a learning rate of $1e^{-6}$. We used the Adam optimizer and divided our dataset into train, validation, and test splits (80-10-10) and reported training performance in Figure 2.

6 Results and Analysis

Model	Train Dataset Size	Pretrained Baseline	Finetuned
GPT2-124M (end token fix)	3800	0.0211	0.0977
GPT2-124M	1900	0.0211	0.0643
GPT2-124M	3800	0.0211	0.0763

Table 1: **General QA task BLEU results.** Quantitative evaluation using the BLEU metric on the general question-answering task. Included are ablations on model size and training dataset size.

Model	Train Dataset Size	Pretrained Baseline	Finetuned
GPT2-124M (end token fix)	3800	0.383	0.653
GPT2-124M	1900	0.383	0.423
GPT2-124M	3800	0.383	0.466

Table 2: **General QA task Embedding Cosine Similarity results.** Quantitative evaluation using the GPT-3 (size 1053) embedding cosine similarity metric on the general question-answering task. Included are ablations on model size and training dataset size.

Model	Train Dataset Size	Pretrained Baseline	Finetuned
GPT2-124M (end token fix)	3800	0.1552	0.1753
GPT2-124M	1900	0.1552	0.1335
GPT2-124M	3800	0.1555	0.1278

Table 3: **General QA task WUPS results.** Quantitative evaluation using the Wu-Palmer Similarity metric on the general question-answering task. Included are ablations on model size and training dataset size.

We present results both on the general QA task (1, 2, 3) and the joint counting task (4) using the test split of our dataset. We use BLEU scores, WU-Palmer Similarity scores, and Cosine similarity with GPT-3 embeddings to evaluate our model performance. Generally, we see that as expected, the fine-tuned model performs much better than the general pre-trained GPT-2 model on the datasets for all metrics. For some metrics like BLEU and WU-Palmer similarity, we were not expecting low magnitude scores. However, we did expect that the use of a single reference answer for the QA pairs would affect the metrics calculation. Broadly, we see that the fine-tuned model has a +400% improvement in the BLEU score, a +180% improvement in cosine similarity, and a +15% improvement in WUPS.

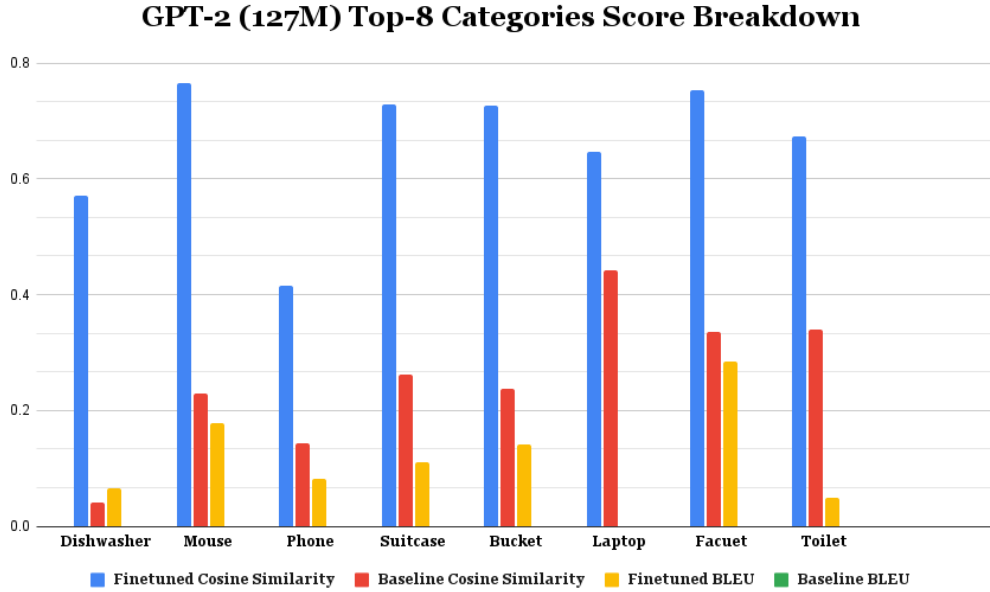


Figure 3: **Top 8 Categories Comparison.** We compare the cosine similarity and BLEU scores for the Top-8 Categories. Note that the BLEU score for the baseline is so low, that it is not visible in this chart.

Model	Error (L1)	Baseline (%)	Finetune (%)
GPT2-124M	0	6.45	50.23
GPT2-124M	1	17.97	76.49
GPT2-124M	2	23.04	85.23

Table 4: **Joint counting task results.** Quantitative evaluation indicating the percentage of times the model correctly outputs the expected joint count. We computed the results with three different error tolerances: zero L1 distance between inferred joint count and ground truth, or exact math, and also 1 and 2 differences between inferred joint count and the ground truth joint count.

Dataset Size. The general QA results indicate that increasing dataset size improves performance after finetuning. In table 1, we see that the performance of row 3 is $0.0763 > 0.0643$. However, the difference in performance did not scale with the increase in the dataset size. Even though the dataset size doubled, the performance increase was close to +15%.

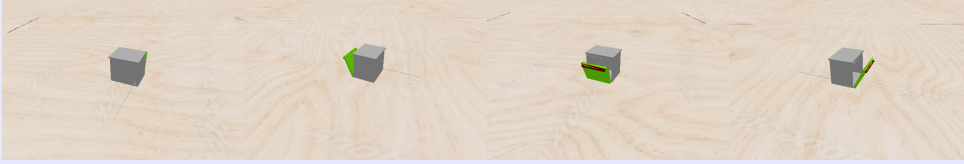
End Token Ablation. One major challenge we faced while finetuning the model was that the model would output a formatted response and then proceed to output hallucinated results. For example, several prompts returned randomly generated URDF content instead of normal text answering the question. This problem was solved by inserting `<|endoftext|>` tags at the end of our training examples to indicate where the model should stop output. We can see that the best-performing model in tables 1, 2, 3 is the GPT-2 124M with the end-token fix. In general, we see a +23% (cosine similarity) to +38% (BLEU) improvement in performance across each of the metrics compared to the regular fine-tuned model.

Training 774M vs. 127M. We fine-tuned two versions of the GPT-2 variant that are public on huggingface. The tables included in the current section are for the 127M model. However, there were several difficulties in training the 774M model including issues of convergence. This also led to issues with "token pruning" or "token sparsity" when the model outputs significantly fewer tokens due to conservative generation learned from a small tuning dataset Kim et al. (2022). We still report the results from our attempt to train the 774M parameter model in the Appendix 5.

Category Breakdown. We see in figure 3 the top-8 best-performing categories for the finetuned model across the cosine similarity and BLEU score metrics. We see that for many of the categories, the cosine similarity is high between the model output and the expected answer, but the BLEU score

is not as high. This highlights the limitation of using the BLEU score metrics when one reference translation or target answer is provided. Note, the top-8 objects have often 1 main hinge joint such as the "Dishwasher", "Suitcase" and "Laptop" where the motion of the joint goes up and down. This could be because the easiest objects to reason about are ones with simple motion and articulation structures. We provide a full table breakdown 6 in the appendix section for each category.

General QA Example



[URDF]

```
<joint name="joint_0" type="revolute"><axis xyz="1 0 0" /><child
link="link_0" /><parent link="link_1" /></joint>
<joint name="joint_1" type="fixed"><child link="link_1" />
<parent link="base" /></joint>
```

[CAPTION] This is a modern freestanding dishwasher shown in a four-angle view, featuring a front-loading design for washing dishes.

[QUESTION] How does the door of the dishwasher open for loading and unloading dishes?

[ANSWER] The dishwasher door opens by pulling it downwards, providing access to the interior racks where dishes can be placed.

Joint Counting Example



[URDF]

```
<joint name="joint\_0" type="prismatic"><axis xyz="0 0 -1" /><..\_0" />
<..\_1" />
. . . <contracted>
<axis xyz="0 0 -1" /><..\_7" /><..\_8" /></joint>
<joint name="joint\_8" type="fixed"><..\_8" /><..\_base" /></joint>
```

[CAPTION] This is a simple rectangular white remote control with minimal buttons, likely designed to operate a specific device or piece of technology. It features a clean and straightforward layout for user-friendly operation.

[QUESTION] How many joints are there in the object?

[ANSWER] There are 9 joints in the object.

Joint Counting Results. The joint counting task results presented in table 4 underscore the model's refined understanding of URDF objects. The improvement in accuracy from the baseline to the fine-tuned model is higher than +40%. For instance, with zero L1 distance, the fine-tuned GPT2-124M model achieves a 50.23% accuracy, over the baseline's 6.45%. We further evaluate the model performance on this task by relaxing the constraint that the model must predict the exact number of joints as in the ground truth. We observe this trend continue with 1 and 2 error tolerances, highlighting

the finetuning process’s efficacy in enhancing the model’s precision in identifying and reasoning about the joints of URDF objects.

Qualitative evaluation of the finetuned model. To better understand the object-language reasoning capabilities of the fine-tuned LLMs, we present a set of qualitative results of the model answering previously unseen questions on objects from the test set. These results highlight the surprisingly strong generalization of the model despite rather limited training data. In the first example of the general Q&A task, we see that there is a clear indication the model understands the movement of the joint that controls the door opening. (Note that the actual image is never given to the model and it is only shown for reference). Additionally, despite no information about the interior rack in the caption, the answer contains information about the rack which is defined in the URDF. For the joint counting model, we see that the model correctly counts the number of joints in the URDF file code presented. This implies that the fine-tuned model can identify the joint keyword in the code despite the large amount of syntax token that surround the definition of the object joints. Moreover, it demonstrates the model’s capability in identifying movable joints rather than fixed joints which are defined with similar syntax. We provide four more output samples like this in the appendix section as well. Each of them demonstrate the model’s ability to determine the motion and articulation of the object.

Evaluation metrics comparison. When we created the dataset, we generated pairs of questions and answers for the model to answer about each of the URDF objects based on GPT-4 vision’s supervision using the rendered image of the object. However, the problem is that for each question, there was only one answer per pair. Therefore, when using the BLEU score metric, the further the model answer is from the actual wording of the single reference, the lower the score. To address this we utilized cosine similarity and WU-Palmer similarity (WUPS) in order to compensate for the issue of matching the direct wording in the reference. WUPS also faces a disadvantage because it uses hypernyms from WordNet Rahutomo et al. (2012) which has a vocabulary limitation. Moreover, if our model hallucinates closer to the end of the response with syntax from the URDF such as "<" and "/" then the overall score comes close to zero. Finally, we use the cosine similarity from the GPT-3 embeddings model using the OpenAI API with a size of 1053. We find that the scores reflect a better representation of the difference in finetuning. However, the main drawback here is that the score itself is not very semantically interpretable, unlike BLEU.

7 Conclusion

In this work, we have demonstrated a novel approach to finetuning natural language processing (NLP) models with Universal Robot Description Format (URDF) files, enabling semantic reasoning about the geometry and articulation of 3D objects. Our method leverages the synergy between textual and geometric data representations, utilizing the large Sapien dataset to parse URDF files and interpret their geometric and dynamic properties. Through the finetuning of pre-trained large language models (LLMs), we improved the model’s capabilities to parse XML, and significantly improved its performance in answering complex queries about objects.

Our experiments revealed the effectiveness of our approach, with the fine-tuned model showcasing an advanced ability to reason through both captions and URDF file contents. The quantitative results highlighted significant improvements in answering questions related to object articulation dynamics and appearance, underscoring the model’s advanced reasoning capabilities. The joint counting task, in particular, emphasized the model’s refined understanding of URDF objects, achieving a significant improvement in accuracy over baseline models.

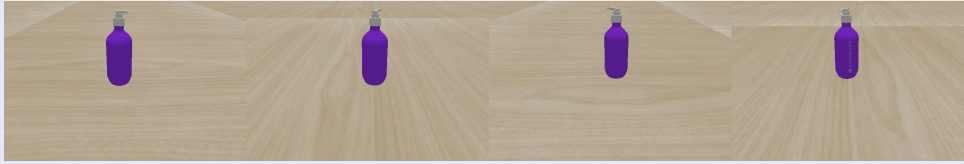
Future work includes training a single model that is finetuned with both the joint counting task and the general QA task, rather than training two separate models. We could also experiment with different QA tasks like asking about the spatial relationships between the axes in the URDF file. Additionally, it would be ideal to fine-tune models with larger context windows to fit more of the original URDF file into the input.

References

Fenia Christopoulou, Gerasimos Lampouras, Milan Gritta, Guchun Zhang, Yinpeng Guo, Zhongqi Li, Qi Zhang, Meng Xiao, Bo Shen, Lin Li, et al. 2022. Pangu-coder: Program synthesis with function-level language modeling. *arXiv preprint arXiv:2207.11280*.

- Lill Maria Gjerde Johannessen, Mathias Hauan Arbo, and Jan Tommy Gravdahl. 2019. Robot dynamics with urdf & casadi. In *2019 7th International Conference on Control, Mechatronics and Automation (ICCMA)*, pages 1–6. IEEE.
- Sehoon Kim, Sheng Shen, David Thorsley, Amir Gholami, Woosuk Kwon, Joseph Hassoun, and Kurt Keutzer. 2022. Learned token pruning for transformers. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 784–794.
- Xiujun Li, Xi Yin, Chunyuan Li, Pengchuan Zhang, Xiaowei Hu, Lei Zhang, Lijuan Wang, Houdong Hu, Li Dong, Furu Wei, Yejin Choi, and Jianfeng Gao. 2020. Oscar: Object-semantics aligned pre-training for vision-language tasks.
- Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. 2022. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097.
- Vadim Lomshakov, Sergey Kovalchuk, Maxim Omelchenko, Sergey Nikolenko, and Artem Aliev. 2023. Fine-tuning large language models for answering programming questions with code snippets. In *International Conference on Computational Science*, pages 171–179. Springer.
- Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks.
- 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.
- Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019a. Language models are unsupervised multitask learners.
- Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019b. Language models are unsupervised multitask learners.
- Faisal Rahutomo, Teruaki Kitasuka, Masayoshi Arisugi, et al. 2012. Semantic cosine similarity. In *The 7th international student conference on advanced science and technology ICAST*, volume 4, page 1. University of Seoul South Korea.
- Walid S Saba. 2023. Stochastic llms do not understand language: Towards symbolic, explainable and ontologically based llms. In *International Conference on Conceptual Modeling*, pages 3–19. Springer.
- Hao Tan and Mohit Bansal. 2019. Lxmert: Learning cross-modality encoder representations from transformers.
- Daniella Tola and Peter Corke. 2023. Understanding urdf: A survey based on user experience. In *2023 IEEE 19th International Conference on Automation Science and Engineering (CASE)*, pages 1–7. IEEE.
- Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, Li Yi, Angel X. Chang, Leonidas J. Guibas, and Hao Su. 2020. SAPIEN: A simulated part-based interactive environment. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

A Appendix



[URDF]

```
<joint name="joint_0" type="continuous"><axis xyz="0 1 0" />
<child link="link_0" /><parent link="link_0_helper" /></joint>
<joint name="joint_1" type="continuous"><axis xyz="0 -1 0" />
<child link="link_1" /><parent link="link_1_helper" /></joint>
<joint name="joint_2" type="fixed"><child link="link_2" />
<parent link="base" /></joint> <joint name="joint_3" type="prismatic">
<axis xyz="0 -1 0" /><child link="link_0_helper" />
<parent link="link_1" /></joint> <joint name="joint_4" type="prismatic">
<axis xyz="0 1 0" /><child link="link_1_helper" />
<parent link="link_2"/></joint>
```

[CAPTION] This appears to be a purple soap or hand sanitizer dispenser bottle with a pump-style top, designed for dispensing liquid product with a push-down mechanism.

[QUESTION] How do you get the soap or sanitizer from this dispenser?

[ANSWER] Press the pump action button which releases the pump from the top of the dispenser.



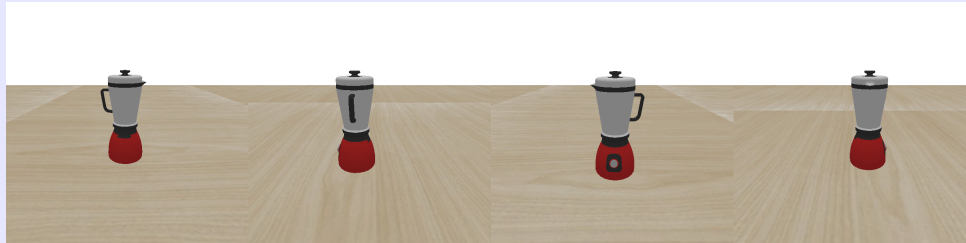
[URDF]

```
<joint name="joint_0" type="revolute"><axis xyz="0 -1 0" />
<child link="link_0" /><parent link="link_1" /></joint>
<joint name="joint_1" type="fixed"><child link="link_1" />
<parent link="base" /></joint>
```

[CAPTION] This is a small, two-drawer bedside table with a white finish and dark brown accents. It features a simple design with rectangular handles on the drawers, which slide out for storage.

[QUESTION] Is it necessary to use any tools to open the door?

[ANSWER] This is not common; typically, the doors can be opened by hand if tools are not present.



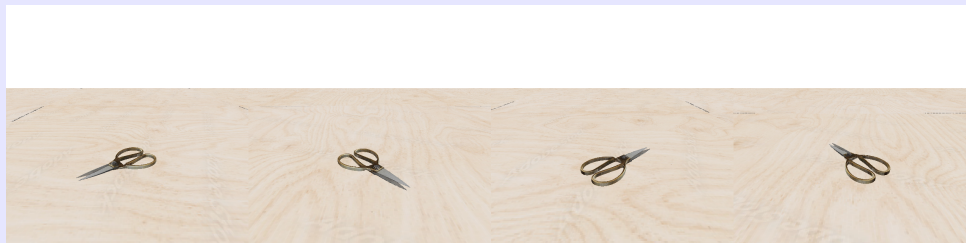
[URDF]

```
<joint name="joint_0" type="prismatic"><axis xyz="0 1 0" />
<child link="link_0" /><parent link="link_4" /></joint>
<joint name="joint_1" type="prismatic"><axis xyz="0 1 0" />
<child link="link_1" /><parent link="link_0" /></joint>
<joint name="joint_2" type="continuous"><axis xyz="0 -1 0" />
<child link="link_2" /><parent link="link_4" /></joint>
<joint name="joint_3" type="prismatic"><axis xyz="0 -0.25884785
-0.9659181050439845" /><child link="link_3" />
<parent link="link_4" /></joint> <joint name="joint_4"
type="fixed"><child link="link_4" /><parent link="base" /></joint>
```

[CAPTION] This image shows a four-angle view of a moka pot, a stovetop coffee maker that brews coffee by passing boiling water pressurized by steam through ground coffee. It features a lower chamber for water, a filter basket for coffee grounds, and an upper chamber for the finished brew.

[QUESTION] How do you add water to this coffee maker?

[ANSWER] 1. You open the upper lid of the moka pot and add water through a nozzle positioned near the top. 2. You would add water by pushing off the pressure valve at the top of the spout at the top.



[URDF]

```
<joint name="joint_0" type="revolute"><axis xyz="0 1 0" />
<child link="link_0" /><parent link="link_1" /></joint>
<joint name="joint_1" type="fixed"><child link="link_1" />
<parent link="base" /></joint>
```

[CAPTION] This is a pair of scissors with metal blades and ringed handles, typically used for cutting various materials such as paper, cardboard, fabric, or other thin materials.

[QUESTION] Can the scissors be used with either hand?

[ANSWER] Yes, the scissors can be used with either hand for cutting various materials like cardboard, fabric, or other thin materials.

Model	WUPS	BLEU	Cosine Similarity
Baseline GPT2 774M	0.1553	0.0211	0.441
Finetuned GPT2 774M	0.1125	0.0746	0.421

Table 5: **General QA GPT2 774M task results.** Quantitative evaluation using the BLEU, WUPS, cosine similarity for the 774M GPT2 Model. We see in these results that the model performance after being finetuned is worse than the baseline due to token sparsity.

Category	Cosine Fine-tuned / Cosine Base	Cosine Finetuned	Cosine Base	BLEU Finetuned	BLEU Base
Dishwasher	14.090	0.571	0.041	0.066	0.000
Mouse	3.341	0.764	0.229	0.177	0.000
Phone	2.877	0.415	0.144	0.082	0.000
Suitcase	2.772	0.728	0.263	0.110	0.000
Bucket	2.411	0.725	0.237	0.142	0.000
Laptop	2.304	0.647	0.442	0.000	0.000
Facuet	2.244	0.753	0.335	0.284	0.000
Toilet	1.983	0.673	0.340	0.049	0.000
Remote	1.945	0.615	0.227	0.206	0.000
Camera	1.943	0.588	0.305	0.046	0.040
Eyeglasses	1.918	0.843	0.329	0.224	0.000
storage_furniture	1.858	0.737	0.230	0.274	0.000
Stapler	1.843	0.746	0.405	0.058	0.000
chair	1.456	0.692	0.343	0.136	0.000
Trashcan	1.320	0.703	0.533	0.107	0.000
table	1.286	0.452	0.352	0.000	0.000
USB	1.258	0.739	0.588	0.255	0.042
refrigerator	1.149	0.607	0.528	0.000	0.000
CoffeeMachine	1.035	0.568	0.501	0.000	0.000
Window	0.897	0.518	0.578	0.000	0.000
Cart	0.890	0.487	0.547	0.074	0.050

Table 6: **Category Model Performance Breakdown.** A breakdown of the baseline and finetuned model performance for GPT2 127M for each category of URDF files.