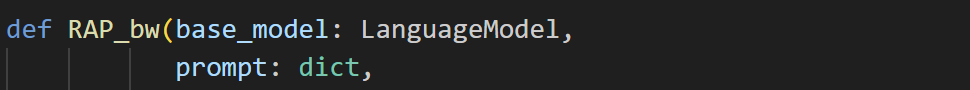
**CSE 576: Topics in NLP**

**Class Project : Team LLM Parrots**

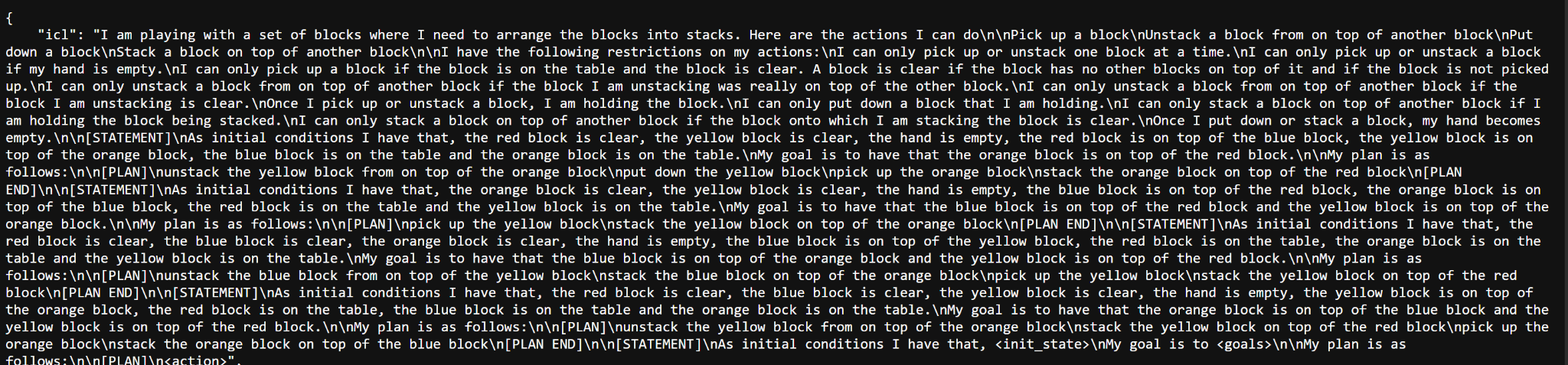
1. **Week 1 (30th October, 2024 - 6th November, 2024):**
   1. Studied and Analyzed the paper to understand the code structure for RAP and LATS. (going through the github repository)
   2. Requested Access for Hugging Face Model: Llama-3.1-8B-instruct.
   3. Attempted to implement RAP algorithm Zero-shot using LLama through demo.ipynb available in the GitHub using the Preprocessed Dataset.

Issues faced while doing the zero shot implementation:

1. The RAP is implemented with a prompt dictionary as input which is accessed from the path ‘examples/CoT/blocksworld/prompts/prompt.json'

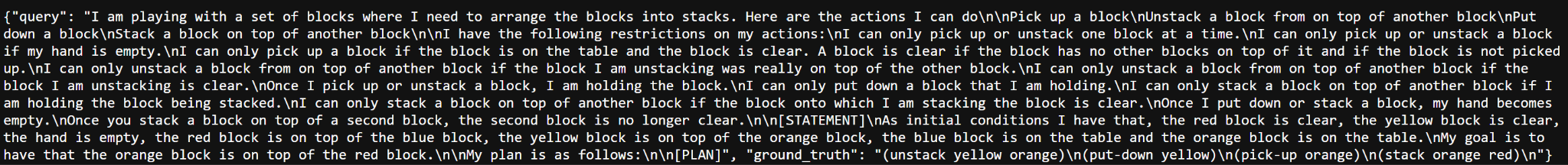


1. The structure of the prompt is as follows:



Highlights of the prompts.json is that we are

1. It has a field like world\_update\_pickup which gives scenarios based on the states and what change needs to be done to do the pickup and the next state after the pickup is done.
2. Similarly, world\_update\_stack, world\_update\_unstack, world\_update\_putdown
3. The amount of information given in the prompt is very very vast and informative and the RAP is implemented using such a huge prompt. It was the main challenge we faced to replicate the RAP repository for a Zero-shot encoding. As in zero-shot encoding we won’t be giving the LLM such information.(A sample query from the zero-shot prompt dataset we need to utilize in this project is given below)



* 1. Failed to reach baseline Zero-shot RAP as the algorithm required prompt dataset

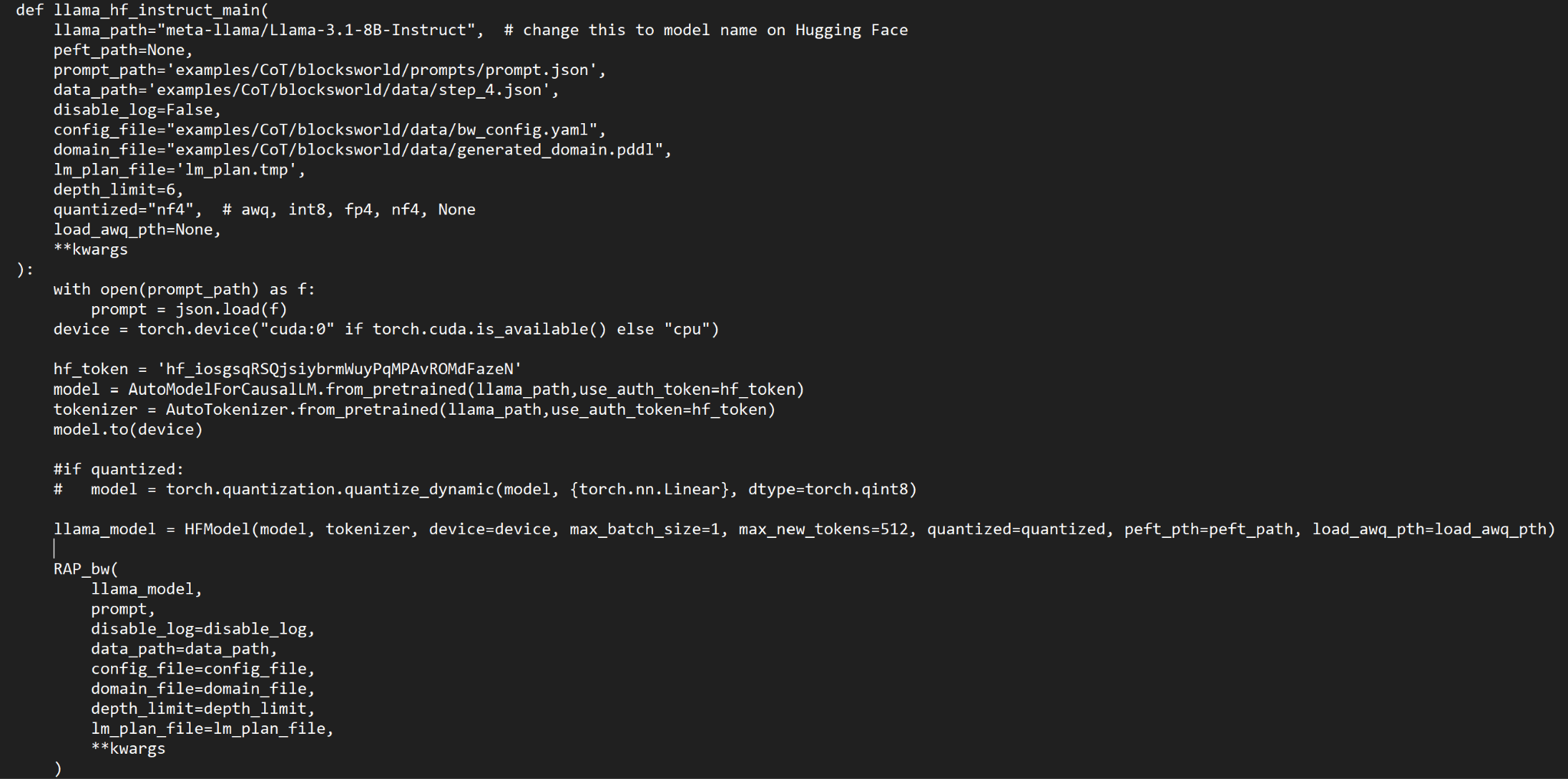
3. Repurposed to Implement RAP and LATS as is, based on how the original code is structured.

1. **Week 2 (7th November, 2024 - November 13th, 2024):**
   1. Code adjustment for RAP in Llama 3.1-8b-Instruct.

We are in-progress with an implementation of the RAP algorithm for the blocksworld domain using Llama-3.1-8B-Instruct.

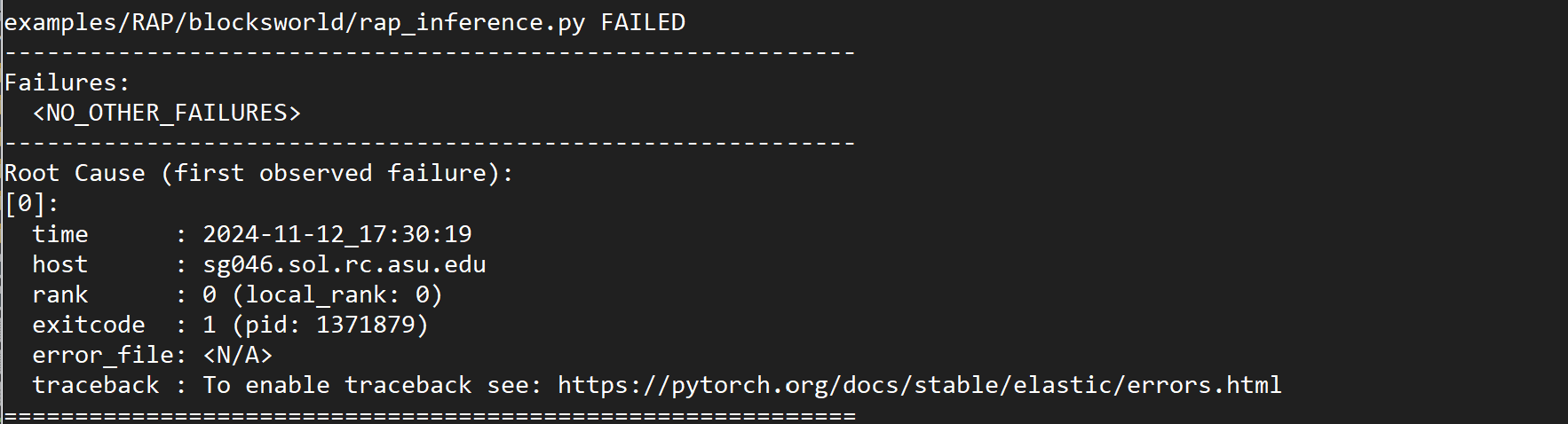
Some roadblocks faced in the implementations:

1. Added the new model in the rap\_interface.py



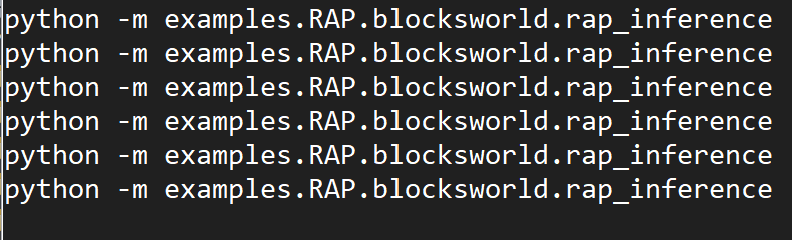
Made changes in the rap\_interface, test\_llama2.sh for the model.

1. We tried the test\_llama2.sh with torch.distributed.run --nproc\_per\_node 1 but into the following errors



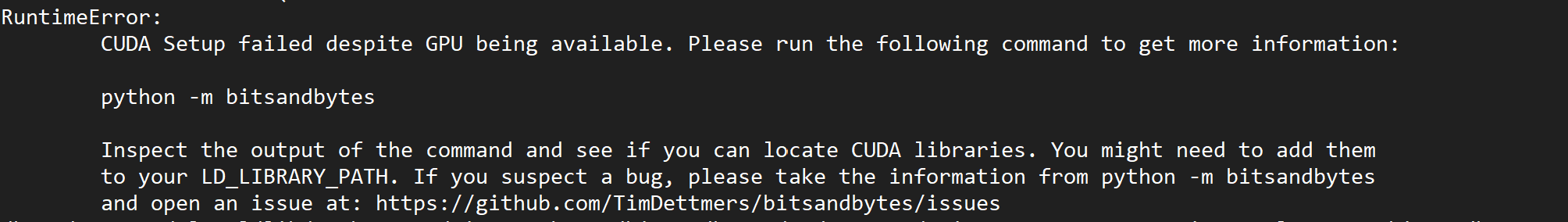
To fix the following issues we ran it without the torch.distributed.run

1. After some modifications and running the python file as script using -m the following changes were made in the test\_llama2.sh file

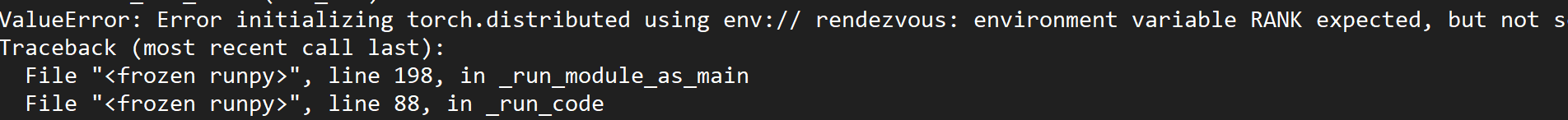


This got rid of the issue in point 1 , as well as the ModuleNotFound errors

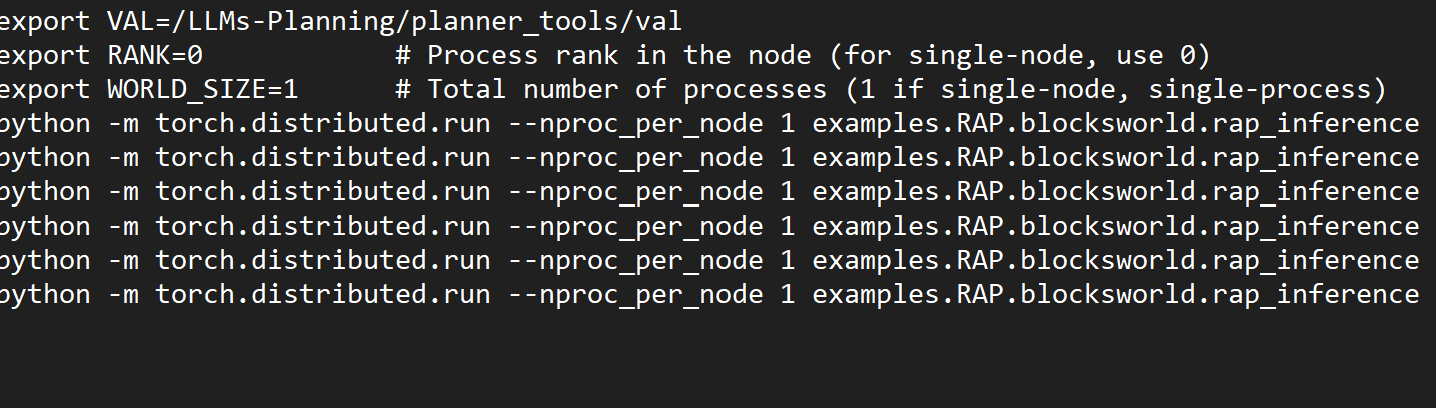
1. The following error was coming after making the above modifications



1. Removing the torch.distributed.run created the following issues.



1. Made some changes in the sh file to fix this error



Still have been getting errors, We haven't received a session after this experimentation.

* 1. Replication of LATS implementation for the given dataset
     1. We are in-progress with an implementation of the LATS algorithm specifically for the blocksworld domain
     2. The major challenges that we faced while implementing the algorithm included state extraction from prompts, an accurate reward metric for simulation, computing new states based on the recommended next action by the LLM
        1. State extraction from prompts

We implemented a python parser to convert the initial state from prompts into an object as follows:

| Clear blocks: {'red', 'yellow'}, Hand: None, Block on top positions: {'red': 'blue', 'yellow': 'orange', 'blue': 'table', 'orange': 'table'} |
| --- |

* + - 1. Reward metric for simulation

The initial thought was to rely on the LLM itself to give us the reward. For this we have a “reward prompt” which takes the current state and asks the LLM to estimate the number of steps required to reach the goal state. We then use the number of states to derive a metric. Ex (100 - n)/100, where n = no. of steps)

Another approach is to derive this metric programmatically by comparing the relative positions of blocks in the current and the goal state. We are still working on this.

* + - 1. Computing new states

We created prompts to derive new states from the current state and the given action by the model. Here, we rely on the model to take us from current to next state. We observed the next states generated by the model are not that accurate. We could also derive the next steps programmatically which would be more accurate but would take away model contribution from the solution.