Experience

I come from a background of statistics and CS so I have experience working with Python and PyTorch. I dont have specific NLP experience but I have taken courses on ML and Computer Vision so I think It will be easier for me to pick up NLP concepts. In the water institute work I am doing a lot of data visualizations using libraries like pandas, numpy, seaborn and matplotllib.

Do As I Can Not As I Say

Framework SayCan designed to bridge the gap between the high-level semantic knowledge of LLMs.
LLMS lack real-world experience and embodiment so they cant make actionable decisions for robots.
What does SayCan do?
LLMs (Say) provide high-level instructions or task goals (“Bring an apple”).
Value functions (Can) measure feasibility of each action based on robots current state and skill set. These affordance function help decide which low level actions lead to successful task completion.
Grounding LLMs through value functions doubles task success rates compared to ungrounded ones
LLMs are effective in generation action sequences and scoring options.
Larger LLMs demonstrate higher task success rates, improving LLM enhances robotic task performance
The decision making process is interpretable since it relies on high-level language tasks and low-level affordances.

Soft Self-Consistency Improves Language Model Agents

Self consistency: sample multiple, diverse reasoning paths and use the generations to select the most consistent answer
Self consistency does not work when tasks have many distinct and valid answers, so selection by voting requires a large number of samples, this can be expensive
Soft self-consistency - gives a continuous score computed from model likelihoods, it allows for selection even when actions are sparsely distributed
Soft-SC outperforms SC with same number of samples

- Given input x generates k solutions. We score the action Yi.
- For each token yi, there is an associated probability P_LM(yi | y<i, x), which represents the likelihood of that token given the previous tokens (y<i) and the input (x). This probability is calculated by the language model
- To score the action y, they aggregate the probabilities of its tokens using a function f, which can be one of the following:
    - **min**: Takes the minimum probability among all tokens.
    - **mean**: Calculates the average probability of all tokens.
    - **product**: Multiplies the probabilities of all tokens together
- aggregated score (score(y)) provides a single value to represent the overall probability of the action y

LLM agents: agent solutions employ chain of thought prompting interleaved with permissible actions within an environment.
Soft SC is compatible with both white box and black box models.
logits = outputs.logits[:, -1, :] append to action logits
You do stack and mean of it
torch.nn.functional.softmax(mean_logits, dim=-1)
action_entropy =-torch.sum(action_probabilities * torch.log(action_probabilities + 1e-10))
action_prob: CrossEntropyLoss()(mean_logits.unsqueeze(0), action_input[:, -1])
action_scores.append((action, action_score.item(), action_entropy.item())