PROJECT PLAN

PROJECT 11: INTEGRATING LARGE LANGUAGE MODELS INTO REINFORCEMENT LEARNING

1 Motivation & Overall Goal

In an RL setting, the agent observes the current state of the environment, and chooses an action to interact with its environment, based on some pre-defined policy. It then receives some reward (or punishment) depending on what new state it ended up with. Initially, the agent typically must guess what actions to perform, as it may know little or nothing of its environment. Iteratively, the agent learns about the environment by traversing states, and updating state/action values based on the reward it receives in each new state. Ultimately, this leads to the agent being able to act intelligently in the environment it is trained in.

However, in a diverse environment with many possible states and actions, this training process is very inefficient, as the initial "guessing phase" takes much longer time as new states and actions are introduced, and thus, exponentially many more combinations of states and actions have to be evaluated by the agent in order to explore the environment.

To combat this effect, we want to introduce environmental domain knowledge to the agent, so it becomes knowledgeable of its environment quicker. This would allow the agent to focus on sensible actions rather than guessing and performing many uninformed ones, thus drastically increasing the speed of the initial learning phase.

Humans are typically good at this, as we have very much domain knowledge of the real-world environment. We know instinctively not to touch very hot surfaces, as we may get burned, and when we see a key next to a door, we intuitively understand that the key is likely meant to lock or unlock the door. Ideally, we would want an RL agent to inherit this instinctive knowledge from us. Thus, a question is raised; how may this be achieved?

With the emergence of state-of-the-art LLMs, an opportunity has been presented to achieve this. These LMs have been trained on vast amounts of human-written text, and are capable of replicating it in conversational way. That is, human knowledge is exists in a written form, and the newest LLMs are able to convey it with high accuracy. Thus, an LLM could perhaps give instructions to an RL agent in the initial learning phase, leading the agent to explore the states that would have been sensible to reach anyway.

The goal of this project is to utilize an LLM to provide instructions to an RL agent in a complex interactive environment with relatively large sets of states and actions. We will attempt this in the Minigrid test environment, with the LLM directly providing the action-choosing policy to the agent. We will compare results from this experiment with corresponding results from using conventional policies. We may also examine the possibility of using an LLM to provide a reward estimation for the agent instead of rewards being predefined.

Given this context, we specifically want to examine the following research questions:

RQ1: To what extent may the use of LLM as RL agent policy increase efficiency in RL training when compared to conventional RL policies, especially when considering the initial learning phase of the agent?

RQ2: How should an LLM be integrated into an RL model in order to increase training efficiency, particularly in the initial part of training, when compared to conventional RL methods? Particularly, should the LLM be used as:

- the policy of the model?
- the reward function of the model?
- both of the above?

2 Project Layout & Plan

2.1 Step-by-step Plan

The following is an ordered list of things to do to achieve the project goal. It is not final, but a guideline of overarching points that are currently considered important in planning the project, and may be subject to change as the project progresses:

1) Read literature relevant to the project (To be done continuously alongside other steps as well)
2) Get familiar with Minigrid, the chosen RL environment
    a. Get familiar with Gymnasium API
3) Create baseline RL model(s) in Minigrid, and test them
    a. Gives us a baseline performance to compare the later, LLM-integrated RL model with
    b. Gives us a modular programming template, with which we can later swap in LLM modules (i.e., policy, reward)
    c. Get further familiarized with Minigrid and Gymnasium (2)
    d. Get experience with RL programming
4) Explore different ways to integrate LLM into RL model
    a. As policy (in what capacity? How directly should it give instructions?
    b. As (part of) reward function
    c. Other ways (guided exploration, plausibly useful behaviors, measurer of interestingness +++)
5) Start integration of LLM API into Minigrid RL model
    a. Use considerations from (4) to decide how LLM is integrated in model architecture
    b. Prompt engineering to get best responses
    c. Encode environment into natural language
    d. Decode LLM response to agent readable action
    e. Think of ways to cache prompts and responses
        i. Restriction on number of requests per second
        ii. May be expensive
6) Test the LLM-integrated model, and report results compared to conventional RL
7) Test different (Minigrid) environments
    a. Focus on diverse environments and generalizability
    b. Optional: Look at other, more challenging environments
        i. E.g., Rouge-Gym, procedurally generated and more complex environment
    c. Look at transferability of training between environments
8) Finish paper, reporting all findings

2.2 Timeline

The following is an outline of when the steps listed above should be performed. It also contains all deadlines from the IN5490 Deliverables timeline:

~~Week 1: 21/8/2023-25/8/2023 - Lecture Week 1~~

~~Week 2: 28/8/2023-1/9/2023~~

Week 3: 4/9/2023-8/9/2023

- Deliver your project plan **by September 6**
- Read literature about RL in general and LLM informed RL
- Start developing RL algorithms in Minigrid
- Start looking for open source LLM models with APIs.

Week 4: 11/9/2023-15/9/2023

- Choose Papers to review
- Make presentation about papers and project
- Continue developing RL models in minigrid
- Begin developing LLM-RL integration

Week 5: 18/9/2023-22/9/2023

- Work on presentation for next week
- Continue working on RL models and LLM integration
- Get preliminary performance metrics of both models

Week 6: 25/9/2023-29/09/2023 - Lecture week 2

- Present paper reviews and briefly outline your project
- Write early draft of paper

Week 7: 2/10/2023-6/10/2023

- **Deliver early-draft of paper**

Week 8: 9/10/2023-13/10/2023

- Work more on code
- Get more performance metrics

Week 9: 16/10/2020-20/10/2020

- Work more on code
- Finish code

Week 10: 23/10/2023-27/10/2023

- Ethics/broader impact statement assignment (draft)
- Polish code, final results
- Work on paper

Week 11: 30/10/2023-3/11/2023

- Work on paper
- **Deliver a draft of your paper on Friday November 3**

Week 12: 6/11/2023-10/11/2023

- Get a paper assigned to review on Monday November 6
- Review other paper
- Work on polishing own paper
- **Review papers assigned to you and hand them in on Friday November 10**!

Week 13: 13/11/2023-17/11/2023 - Lecture Week 3

- **Final presentation**
- **Deliver Final Paper on Friday November 17!**

2.3 Methods/Tools

Because the project is still in its early stages, this section is also subject to change. In particular, we haven't decided on which LLM to utilize yet, and have put this as a task to be performed in week 3.

Regarding methods, we plan on first developing a "baseline" RL model that works in Minigrid, using some conventional RL method(s), like Q-learning and/or something sophisticated like PPO. We intend to program this modularly, so that the policy and/or reward functions can be easily replaced by alternative ones.

Once the RL model is set up, we want to swap the policy and/or reward functions for an LLM-based alternative. For the policy, we are considering both an LLM directly controlling agent manipulation of the environment, and the LLM making several suggestions for the agent, and the agent choosing one, as discussed in some of the literature for the project. For the reward function, a natural application of an LLM could be to have the LLM express what an appropriate reward for a given action should be. For all these cases, we believe it necessary to develop some algorithms to translate input and output from both the agent and the LLM, so they can communicate.

We intend to use Minigrid as an environment to test our RL model in. Minigrid is part of the Gymnasium API. We are currently investigating free open source LLMs to use for our project. Should it be possible, we are also considering using GPT 3.5 as an option. Another option is to run Llama 2 locally. Should we choose to use more advanced RL methods as a baseline for comparison, we plan to use Torch RL, which is part of the PyTorch library.

3 Team Meetup Planning & Responsibility Delegation

All participants on the team plan on generally being at IFI most workdays for working on all courses this semester, including IN5490, except when this conflicts with lectures, group lessons etc. We share many courses and generally plan on working on these together and will thus naturally focus on working on the project during this time as well.

For meetings with our supervisor, we think a weekly update meeting of 20-30 minutes is beneficial and propose finding a specific time during the week when this fits for everyone.

Considering that we will spend much time together physically when working on the project, we think it is natural to delegate responsibility for specific tasks as they appear. Many of the tasks, like programming and reading literature, must be performed by all participants regardless, and may often require collaboration.