# Building LLM Reasoners Final Project

**Proposal Due: March 27, 11:59pm**
**Check-ins: April 17, 11:59pm**
**Presentations: May 1**
**Final Report Due: TBD**

**Collaboration** You are free to work on this project in teams of two or three. Two-person projects can be less ambitious but should not be less complete: a half-implemented system does not make a good project outcome. All partners should contribute equally to the submission, and all partners will receive the same grade for it. You may collaborate with people from outside the course as well in case you're also using this final project for another course. You are also free to discuss your project with others in the course, though only the people on your team should contribute to the actual implementation/experimentation involved. Any external resources used must be clearly cited.

**Combining with other final projects** You are allowed to combine this project with your research or projects from other courses. However, your project must still involve concepts from this course! You are allowed to apply these models to data that isn't language data provided that it has some interesting language-like structure (e.g., genomics data, time-series data, etc.). Investigating feedforward neural network architectures on MNIST would *not* be an acceptable course project.

## Overview

This project is an independently-conducted study of some aspect of LLMs and reasoning tasks (broadly construed). You have two options. The first is to pursue original research on an NLP problem, and the second is to attempt to reproduce results from a prior paper.

**Original Research** There are several possible approaches here. You might do a more engineering-style project: pick a task and a dataset, design or expand on some model or algorithm, and try to get good results, similar to what you were doing in the other projects in the course. You can also do a more analytical project: pick some problem and try to characterize it in greater depth. What does the data tell us? What does this tell us about the state of LLM reasoners? What can interpretation techniques or other focused evaluation measures tell us about how models are doing? Most contributions appropriate for an NLP/ML conference or workshop paper are in scope here.

Your end goal for this option shouldn't be just reimplementing what others have done. However, implementing someone else's model or downloading and running an existing model are great first steps and might end up getting you most of the way there. Implementing a couple of approaches in order to gain some insight from comparing them can be a good project (somewhere between research and reproduction, and that's fine!). **For projects in this area, you should start with literature search and include that in your project proposal to make sure you're not missing relevant prior systems.**

This project is *not* graded on how good your results are, as long as you can convincingly show that you've done something: asked an interesting question, conducted experiments to investigate that question, collected high-quality data, etc. Start with baby steps rather than implementing your full approach from scratch: build baselines and improve them in a direction that will eventually take you towards your full approach. You should think these steps through in your proposal.

**Reproduction** The goals here follow those of the ML Reproducibility Challenge.[1] You should pick a prior paper (it can be from any year, although more recent and less-tested methods are more likely to yield surprising results) and evaluate how well you can reproduce the results of the paper. This involves several steps:

1. Figure out what results you want to reproduce.
2. Figure out what code is available, and see if you can get it running easily. If you can get it running in 30 minutes, that will change the scope of your project compared to code that might take hours to resolve or intervention from the authors.
3. Decide what you want to focus on in your reproduction. To quote from the challenge: *Just re-running code is not a reproducibility study, and you need to approach any code with critical thinking and verify it does what is described in the paper and that these are sufficient to support the conclusions of the papers. Consider designing and running unit tests on the code to verify it works well and as described. Alternately, the methods presented can also be fully re-implemented according to the description in the paper.*

**We will hold reproducibility papers to a high standard!** In particular, you should do exploration and experimentation on par with what's expected from the original research option. Don't pick a paper where you can reproduce the network in 20 lines of PyTorch, tune hyperparameters, put your results in a table, and declare victory. Try to have an interesting question that you can answer in a deep way.

## Deliverables

**Proposal (5 points)** You should turn in a **one page proposal** on the proposal due date. This proposal should outline what problem you want to address or what paper you're reproducing, what dataset(s) you plan to use, prior work, and a rough plan for how you will pursue the project. While you don't need a full related work section, you should mention the most relevant prior work you've found and state how your project relates to it. The course staff will then provide feedback and guidance on the direction to maximize the project's change of succeeding.
**Grading:** 5 points for turning in a proposal meeting a minimum level of coherence and quality. You are not evaluated on how good the idea is—this is a stage to get feedback and refine things.

**Check-in (5 points)** You should turn in a check-in on the check-in due date. This should probably be at least 2-3 pages in length and look like a fleshed out version of the proposal, building towards your final project. You should respond to feedback from the proposal stage, include a more detailed discussion of related work, and most critically, *have some kind of preliminary results for your approach.* Think about what you can do to make sure you have something to show at this stage, even if it's more unit test-style validation that your code is correct.
**Grading:** You will lose points on the check-in if we perceive you have not dedicated time to flesh out the project beyond the proposal and take a first stab at the work on it.

**Final Report (80 points)** The primary deliverable is a paper written in the style of a NeurIPS/ACL/etc. conference submission. It should begin with an abstract and introduction, clearly describe the proposed idea or proposed reproduction, present technical details, give results, compare to baselines, provide analysis

[1] `https://paperswithcode.com/rc2020/task`

and discussion of the results, and cite sources throughout (you'll probably want to cite at least 5-10 papers depending on how broad your topic is).

Your paper might be around 6-8 pages excluding references. Don't treat these as hard page requirements or limits. If you have lots of analysis and discussion yielding full-page figures or are trying something more ambitious, your paper might be longer; if you're implementing something complex but succinctly described, your paper might be shorter.

Critically, you should approach the work in such a way that success isn't all-or-nothing. You should be able to show results, describe some successes, and analyze why things worked or didn't work beyond "my code errored out." Think about structuring your proposal in a few phases (like the projects) so that even if everything you set out to do isn't successful, you've at least gotten something working, run some experiments, and gotten some kind of results to report.

**Grading:** We grade projects starting from a "base score" of 70/80 for projects that are executed according to the specification laid out here. This score will then be moved up or down based on the following criteria:

- **Scope**: Was the scope appropriate for the project? You will gain or lose points depending on our subjective assessment of the overall work done and whether it fits the charge you were given.
- **Clarity/Writing**: Your paper should clearly convey a core idea/hypothesis, describe how you tested it/what you built, and situate it with respect to related work. Doing a thorough job of this will raise your grade. Your grade will be lowered if you fail to explain what you did in a clear fashion or do not present results in a coherent way.
- **Implementation/Soundness**: Is the idea technically sound? Do you describe what seems like a convincing implementation? Is the experimental design correct? You may lose points here if
- **Results/Analysis** Whether the results are positive or negative, try to motivate them by providing examples and analysis. If things worked, what error classes are reduced? If things didn't work, why might that be? What aspects of the data/model might not be right? If you're writing a paper that revolves around building a system, you should try to report results for a baseline from the literature, your own baseline, your best model, and possibly results of ablation experiments. If you're doing a reproduction, try to be as thorough as you think is appropriate.

**Final Presentation (10 points)** During the last class, every group will give a 3-minute presentation on their project (depending on the number of groups). This presentation should state the problem, describe the methodology used, and give highlights of the results. Because the project reports won't have been due yet, these results might be preliminary, but should be nonzero.

You will want to keep this presentation focused on high-level aspects of your approach, methodology, and results/examples of the output.

**Grading:** The final presentation should be clear and fit within the allotted time limit. It should describe your methodology, related prior work, and preliminary results or analysis.

## Choosing a Topic

There are a large number of possible topics, following the lectures in the course. A non-exhaustive list is:

1. Capabilities: looking at state-of-the-art reasoners and agents, evaluating them, understanding their successes and failures, and (potentially) improving them. Analysis alone could be a fine project (or building a new dataset or new tasks).

2. Science: if you want to understand RL methods, training, inference, GPUs, etc., you can take small-scale settings like the assignments and explore questions related to these there. (We suggest small-scale because of course doing any training on top of 7B+ models requires computational resources beyond what we've made available to you in the course, but it's great if you have those and can do it yourself!)
3. Interpretability: analysis of model internals, mechanistic interpretability, deeper understanding of how reasoning models work.
4. Safety: benchmarking and evaluation, developing guardrails, and other projects relate dto AI safety.

You have broad freedom to choose what you want to do, but finding a good project can be tricky, so we provide some more guidance here.

General features of good projects:

1. Answers a scientific question with small LLMs OR builds on frontier models. If you want to train models, you should be able to train very small LLMs to answer your question. If you want to prompt models or design reasoning harnesses, you should use the strongest frontier models (or strong open-source models like the largest Qwen or gpt-oss models) to obtain good performance.
2. Multiple success "modes": you can define an initial goal which is quite likely achievable, then have some stretch goals to aim for.
3. Enough analysis/experiments to do: The project is not just all-or-nothing success or failure on some target dataset, but you can provide interesting analysis of why the results are how they are.

General features to avoid:

1. Projects that primarily involve domain engineering and have limited scope for technical novelty. For instance, developing an agent for betting markets has limited scope for interesting work regarding reasoning or LLMs themselves, and mostly involves applying LLMs to that domain.
2. High computational requirements: projects involving fine-tuning, GRPO on large models and large datasets, etc., unless you're able to bring your own compute resources to the project.

## Computational Resources Available

Compute is available through the other channels discussed in the course. You can also use cloud resources such as Runpod or GCP, which may have fairly cheap GPUs that can serve your purposes and which sometimes offer credits.

Unfortunately, we are not able to provide sponsored access to closed-source models. However, many projects may be doable for only modest cost (less than $20 in expenditures).

Do not wait until late in the project to sort access issues out, as availability of all of these services is subject to change.

## Submission

You should submit your final report in a single PDF on Gradescope. You should upload your code on Brightspace **for documentary purposes. We do not need to see large data or every piece of code you produced,** just the most important stuff.

**Slip Days** Slip days may not be used for any component of this project.