NOTE: The planning in this doc is based on my own research & experiences, but *I am very happy to change any part of it (including any parts of the methodology) based on what mentees think is best*. This is really meant to be a first draft & **not a finalized plan**.

Motivation

- The current paradigm for aligning LLMs relies heavily on safety posttraining. However, posttraining pipelines tend to encounter various difficulties, such as instability, reward hacking, & alignment taxes.
- Traditionally, the field has viewed the pretraining corpus as a neutral slate, focusing on tweaking posttraining algorithms to solve these issues. However, emerging research demonstrates that **optimizing pretraining is important for obtaining the best safety posttraining alignment** (Mizrahi et al).
- This project focuses on **how best we can optimize pretraining data curation to maximize posttraining safety**. By jointly optimizing pretraining and posttraining, we can save costs on posttraining while achieving better safety.
- Prior work has shown that optimizing posttraining to maximize **pure quality often leads to brittle models which vary in performance** (Moskovitz et al).
- Prior work on data curation for posttraining in particular has shown that **the best way to optimize data for posttraining is to use data that promotes stability** (Zhang et al, Deng et al).
- Thus, this project focuses specifically on how to optimize pretraining data curation for **maximum posttraining stability**. To do this, this project will use an **influence approximation (TRAK) to identify which pretraining documents** contribute to instability.
- By identifying and filtering out the pretraining data that causes downstream reward variance, we can curate a stabilizing pretraining corpus that **makes safety posttraining more effective**.

Research Question

How can we optimize pretraining data curation to maximize safety posttraining effectiveness?

Related Work

- Mizrahi et al 1: prove that optimizing pretraining data for posttraining is important
- Mizrahi et al 2: show the importance of multiprompt evaluation for evaluating true model capability
- Moskovitz et al: show that optimizing for pure “quality” usually doesn’t lead to best performance due to conflicting objectives, which leads to instability

- Rafailov et al: show that overoptimization & reward hacking can occur for offline posttraining as well
- Zhang et al: develop an approach to curate less noisy preference data for PPO (DORM)
- Deng et al: develop an approach to curate less noisy preference data for DPO (BeeS)
    - Also show that noisy data leads to a form of model degradation called parameter shrinkage
- Park et al: developed TRAK, a popular data attribution method

Methodology

- Data Curation
    - Curate pretraining & alignment corpora
    - Pretraining corpus: Standard pretraining mix (e.g. FineWeb).
    - Alignment corpus: Standard preference dataset (e.g. UltraFeedback), augmented with synthetically generated paraphrases for each prompt to create "prompt families." This allows us to measure variance (stability) across semantically identical instructions.
- Baseline Pretraining
    - Train policy model (e.g. Qwen 3.5) on full pretraining corpus
- Variance Auditing
    - Generate responses using the policy model across the perturbed "prompt families."
    - Score all responses using an off-the-shelf reward model/RM (e.g. Skywork Reward V2) to isolate the prompt families that exhibit high reward variance (e.g. Prompt A answer gets a high reward, Paraphrase B answer gets a low reward).
- Influence Attribution
    - To find the source of the instability, compute the parameter gradient for the high-reward prompt and subtract the gradient of the low-reward paraphrase. This creates a single "variance vector."
    - Then, use TRAK to compute the dot product between this variance vector and the representations of the pretraining documents. Documents with a high positive score are mathematically identified as the ones pulling the parameters in a way that causes the model's unstable behavior.
- Curation & Retraining
    - Filter the top destabilizing documents from the pretraining corpus, & retrain policy model. Optionally repeat cycle
- Posttraining
    - Align via GRPO
- Evaluation
    - Stability: Reduction in reward variance across perturbed prompt families on held-out evaluation sets, measured by the RM
    - Capability: Evaluation on safety benchmarks

Risk Mitigation

- Data Curation
    - **Expected Outcome**: Curate pretraining & alignment corpora
    - Small Possible Risk
        - The generated paraphrases are too similar to each other or lose the semantic meaning of the original prompt.
        - *What this tells us:* The LLM used for generation needs a stricter system prompt.
        - *Next Steps:* Adjust the generation prompt to enforce higher lexical diversity while strictly penalizing semantic drift.
    - Large Possible Risk
        - The generated paraphrases don't capture realistic variance
        - *Affects*: Variance Auditing & Evaluation
        - *Contingency Plan*: Rely on an existing dataset that naturally contains high prompt diversity (e.g., an open-source dataset mapping different user queries to the same intent).
        - *Simpler Way*: Use a small set of rule-based, hard-coded templates (e.g., swapping "Explain X" with "What is X") instead of computationally expensive LLM generation.
        - *Small Test*: Manually review a random sample of 100 generated prompt families to ensure semantic equivalence & sufficient lexical variance (e.g. using Vendi score)

- Baseline Pretraining
    - **Expected Outcome:** Train policy model
    - Small Possible Risk
        - The baseline training run experiences unstable loss spikes
        - *What this tells us*: The pretraining mix contains bad data
        - *Next Steps*: Analyze the corpus for anomalies
    - Large Possible Risk
        - Full pretraining exhausts compute budget
        - *Affects*: All subsequent steps.
        - *Contingency Plan*: Instead of doing a full pretraining run from scratch, utilize an off-the-shelf base model checkpoint as the "baseline" and assume it represents standard pretraining.
        - *Simpler Way*: Perform continued pretraining (tuning) on a subset of data rather than a from-scratch initialization.
        - *Small Test*: Monitor basic perplexity on a validation set during the first few hundred steps to ensure the model is learning normally.

- Variance Auditing
    - **Expected Outcome**: Score policy model responses to isolate the prompt families that exhibit high reward variance
    - Small Possible Risk
        - Almost all prompt families show low variance.
        - *What this tells us*: The RM might be overly smooth/insensitive, or the baseline model is unusually robust.
        - *Next Steps*: Test a different RM or dramatically increase the complexity of the paraphrases.
    - Large Possible Risk
        - The RM used to measure variance possesses its own inherent biases or instabilities. If the RM's scoring is erratic, the downstream variance vectors will be flawed.
        - *Affects*: Influence Attribution (“garbage in, garbage out”)
        - *Contingency Plan*: Aggressively test RMs first to choose the best one, or swap to a simpler rule-based heuristic (e.g., measuring variance in response length or n-gram overlap) to identify instability.
        - *Simpler Way*: Use a small LM-as-a-judge to flag only severe contradictions in responses, rather than relying on precise scalar RM scores.
        - *Small Test*: Give multiple RMs known good/bad pairs & known identical pairs, choose RM w/ most stable & trustworthy scalar signals

- Influence Attribution
    - **Expected Outcome:** Use TRAK to compute the dot product between the variance vector & the representations of the pretraining documents to identify the ones causing unstable behavior.
    - Small Possible Risk
        - TRAK returns highly noisy or overly localized document attributions.
        - *What this tells us*: The approximation is too granular, flagging random anomalies rather than systemic issues.
        - *Next Steps*: Cluster the highly attributed documents to find semantic or syntactic patterns, & filter at the cluster level.
    - Large Possible Risk
        - Calculating gradient differences doesn’t work because the vector space is too high-dimensional, causing the variance vector to essentially be noise.
        - *Affects*: Curation & Retraining
        - *Contingency Plan*: Instead of full-parameter gradients, compute gradients only for the final few layers of the model
        - *Simpler Way*: Perform standard data ablation (leave-batch-out) on a micro-scale to isolate noisy batches instead of using TRAK.
        - *Small Test*: Create a tiny dataset with an intentionally injected "poisoned" document. Run the TRAK pipeline to see if it successfully flags that specific document.

- Curation & Retraining
    - **Expected Outcome:** Filter the top destabilizing documents & retrain
    - Small Possible Risk
        - The retrained model shows no improvement in stability.
        - *What this tells us*: The documents filtered were not the true root cause, or the filtering threshold was too conservative.
        - *Next Steps*: Increase the filtering threshold or redefine how the variance vector is constructed.
    - Large Possible Risk
        - Removing the destabilizing documents inadvertently removes factual knowledge, degrading the model's general capabilities.
        - *Affects*: Posttraining & Evaluation
        - *Contingency Plan*: Do not remove the documents entirely; instead, simply downweight them during the loss calculation.
        - *Simpler Way*: Filter based on simple heuristics (e.g., removing documents with highly repeated n-grams) derived from looking at the TRAK clusters, rather than relying purely on the mathematical score.
        - *Small Test*: Train a tiny model (e.g., 100M parameters) on the filtered vs. unfiltered subsets & plot their validation loss curves to ensure the filtered dataset doesn't cause catastrophic forgetting.

- Posttraining
    - **Expected Outcome:** Align via GRPO
    - Small Possible Risk
        - GRPO encounters reward hacking.
        - *What this tells us*: The pretraining corpus might still contain exploitable artifacts, or the KL penalty is too low
        - *Next Steps*: Adjust the GRPOhyperparameters or switch alignment algorithms.
    - Large Possible Risk
        - GRPO instability prevents convergence regardless of the pretraining data quality, obscuring the results of the experiment.
        - *Affects*: Evaluation
        - *Contingency Plan*: Switch to GRPO or DPO, which are generally more stable & less hyperparameter-sensitive.
        - *Simpler Way*: Use a better, modern variant of GRPO.
        - *Theoretical Test*: Run a brief GRPOsweep on a proven, stable base model to verify that the alignment pipeline itself is sound before applying it to the custom retrained model.

- Evaluation
    - **Expected Outcome:** Reduction in reward variance & increased performance on safety benchmarks
    - Small Possible Risk
        - Stability improves, but standard capabilities drop.
        - *What this tells us:* There is a trade off or "alignment tax" inherent in the filtering process.
        - *Next Steps:* Quantify the trade off & analyze exactly which capabilities were lost to refine future data curation.
    - Large Possible Risk
        - The RM used for evaluation doesn't correlate with actual human judgment of variance.
        - *Affects*: Final conclusions
        - *Contingency Plan*: Allocate a budget for human evaluation on a small, randomized subset of the held-out prompt families.
        - *Simpler Way*: Use an automated, multiprompt evaluation script using a small LM as an evaluator.
        - *Small Test:* Validate the evaluation framework first by running it on existing open-source models known for high & low stability.

Timeline

- *Week 1: Onboarding & Tooling*
  - Get familiar with tooling.
  - Set up data pretraining mix.
  - Set up pretraining pipeline.
- *Week 2: Evaluation Framework & Reward Model Selection*
  - Set up evaluation suite for measuring reward variance.
  - Test different reward models to find the one with the most stable signals.
- *Week 3: Variance Auditing Setup*
  - Generate the paraphrase "prompt families".
  - Write scripts to automate the policy model generating responses across prompt families.
  - Build variance auditing pipeline.
- *Week 4: TRAK & Posttraining Setup*
  - Build the TRAK and gradient difference calculation pipeline.
  - Set up the GRPO pipeline.
- *Week 5: Initial Pretraining & Filtering*
  - Do the baseline pretraining run on the full corpus.
  - Run the variance audit & TRAK attribution on trained model
  - Extract the dot products to mathematically identify the destabilizing documents.
  - Filter the pretraining corpus.
- *Week 6: Initial Posttraining*
  - Retrain the policy model on the curated data.
  - Align the retrained model via GRPO
  - Run the base, standard posttrained model, & optimized posttrained models through the evaluation framework
- *Weeks 7-8: Adjustment*
  - Adjust pipeline based on any unexpected results.
- *Weeks 9-10: Ablation Studies*
  - Conduct ablation studies to better understand pipeline
- *Week 11: Documentation*
  - Document findings
- *Week 12: Buffer Time*
  - Buffer time in case things take longer than expected

Again, this isn't a finalized plan, & I'm very happy to change any part of this doc based on what mentees think is best.