AISI | AI SECURITY INSTITUTE

AISI Protocol for Elicitation Experiments

The following is a **structured protocol for elicitation experiments**, designed to facilitate the adoption of best practices.

Following this checklist step-by-step ensures rigor in current experiments but also contributes to the broader goal of continuously improving testing methodologies. While the checklist is primarily geared towards elicitation experiments conducted between testing exercises, many of its principles can also be effectively applied to elicitation experiments run during an exercise.

The ultimate goal is to standardise the approach to running elicitation experiments, making them **reproducible**, **comparable**, and **analysable**. By doing so, we can collectively build a robust body of knowledge about what works and what doesn't. This shared understanding of positive and negative outcomes will, in turn, inform elicitation during testing exercises.

Experimental setup

☐**I have set a compute budget for this elicitation experiment**.

What is a reasonable amount?

It is difficult to provide a single number or range here, as computational requirements vary largely across tasks and risk domains. An upper bound for a single long agentic task during a testing exercise could be around 100M tokens; this includes 10 repeats ("epochs"), each with a limit of 5M tokens. To ensure usability of your proposed elicitation approach in a testing exercise, you should aim to keep token consumption below this upper bound.

☐**I have identified at least one appropriate baseline.**

Why do we want a baseline?

A baseline provides a standard reference point to measure the performance of a new elicitation techniques or setup. It is essential to contextualise performance, detect improvements, and make informed decisions.

What is a reasonable baseline to use?

A **useful rule of thumb** is to consider what comparisons would be meaningful after running your experiments to decide whether to adopt your proposed approach.

If I am testing a new technique (e.g., a new tool) or conducting a grid search over parameters (e.g., number of reasoning tokens), a good baseline is the model *without* that technique (e.g., without the tool, or with reasoning off).

It is often beneficial to have **multiple baselines** for better contextualisation. For example, if setup X is currently the best-tested option, including X as a baseline will provide valuable context.

☐I have identified proper splits of the evaluation dataset for my experiments.

Note: these splits apply to elicitation experiments involving no model training. This is the most frequent case. Fine-tuning experiments require an additional validation set; see next point.

How to define these three splits?

1. **Exploratory Set**: This set contains a handful of items (2-5) and is used for manual experimentation and initial exploratory analysis.
2. **Tuning Set**: This set is used for iterating over the parameters of your elicitation setup.
3. **Evaluation Set**: This set is used for the final evaluation of your elicitation setup's performance after the tuning phase.

Note: these three sets should be *disjoint* (i.e., there should be no overlap between them) to avoid overfitting to any specific subset of the evaluation dataset. Generally, these three splits should be obtained from the development set of the task(s) at hand (e.g., the "Cyber dev set"). I should *not* look at the test set (e.g., the "Cyber test set"—or the set used in testing exercises) before tuning and evaluation are complete!

☐If I am fine-tuning a model, I have followed a classic ML setting, where the model is trained on a training set and validated on a validation set (e.g., two disjoint portions of the tuning set above). I have made sure that there is no overlap between the training (fine-tuning) set, the validation set, and the evaluation set.

☐ **I have confirmed that I have enough samples to conduct a conclusive study.**

How many samples are enough?

These are the steps required to determine the necessary sample size for your study to have sufficient statistical power:

1. **Identify Your Analysis Type**: Start by thinking of the exact analysis you want to run. For example, is it a t-test between condition A and condition B, are you measuring Pearson correlation between A and B, or are you running an ANOVA with multiple groups?
2. **Conduct a Pilot Study**: Run a pilot study on a small portion of the dataset. Conduct the analysis on the pilot data and extract the effect size (e.g., difference in the means divided by the pooled standard deviation if you do a t-test, or Cohen's *d*), the coefficient *r* if you measure correlation, etc.
3. **Estimate Sample Size Using G*Power**: Download the analysis software [G*Power] (https://www.psychologie.hhu.de/arbeitsgruppen/allgemeine-psychologie-und-arbeitspsychologie/gpower). Select the type of analysis (e.g., correlation), type the effect size, and ask for the sample size required. This will give you the sample size needed to detect your effect size with a significance level (α, commonly set to 0.05) and power (commonly set to 0.8).

Alternative Approaches:

If you cannot run a pilot study: Look at previous literature, find studies that conducted similar experiments, and use their effect size for the sample size estimation.

If you have a more complicated analysis: G*Power works for simple analyses like the ones mentioned above (you will see a drop-down list on the GUI). For more complicated analyses, you will need to run a power simulation yourself:

1. Choose a sample size.
2. Simulate many datasets with your effect size.
3. Analyse each simulated dataset with your analysis.
4. Count how many times you correctly detected the effect (power = number of significant results / total number).
5. Repeat with different sample sizes until you achieve a power of 0.80.

Eval logs

Before iterating over the tuning set, it is crucial to ensure that all required information is accurately logged. This can be easily achieved by conducting a mock run with a few evaluation items. In particular:

☐**I have verified that the model details and elicitation setup are fully logged in the eval metadata.** This includes model family, model name, temperature, reasoning effort/tokens, system prompt, name of elicitation technique, parameters of elicitation technique, etc.—*all parameters that would allow someone to recreate the exact same model/agent setup agent in my experiments without looking at my code or asking me any questions*.

☐**I have verified that all evaluation data details are fully logged in the eval metadata**. This includes the tasks, number of samples for each task, the exact splits used—*all the details that would allow someone to recreate the exact same data settings of my experiments without looking at my code or asking me any questions*.

Elicitation

When working with a new model or task, we should start by understanding and addressing any failure modes. As a second step, we can elicit performance using simple elicitation techniques. Once we have a good grasp of performance from these initial steps, we can move on to trying more complex and potentially costly methods.

Detect and fix egregious failures

☐**I have looked at transcripts from cases where the model fails the task being evaluated, I fixed egregious failure modes, and I am confident that I know the major reasons for the remaining failures.**

☐**I have verified that the model is not failing due to refusals.** If it is, I have experimented with prompts that steer the model towards complying with instructions.

☐**I have verified that prompt and chat messages are properly formatted (e.g., keeping indentation and paragraph structure intact).**

☐**I have verified that the model is not producing the correct output but using the wrong format**. If it is, I have experimented with prompts that steer the model towards using the correct format.

☐**I have verified that the model is not failing due to confusion about the task. If it is, I have experimented with clarifying the prompt.**

☐**I have verified that in cases where the model is marked as failing the eval, it is actually failing to accomplish the task and is not due to a mistake in scoring.**

What if there are mistakes in scoring?

These mistakes can occur, for example, when questions have multiple correct answers or if there is an issue with the scorer. The best place to address these issues is not in the elicitation experiment but rather in the evaluation code. Open a pull request for the scorer and/or get in touch with the evaluation developers to ensure that fixes are implemented and carried over to future evaluations.

Tool use

If the task relies on tool use, computer use, or other such capabilities:

☐**I have made sure that the model has access to all the tools necessary for solving the task.**

☐**I have verified that the model uses the correct format (e.g., that it doesn't incorrectly escape newlines) when calling tools.** This is a failure mode to look out for both in failed tool calls and "successful" tool calls where the arguments are incorrect.

☐**I have verified that the tools are being used appropriately and not in unintended or counterproductive ways.**

☐**If a tool appears counterproductive, I have experimented with removing the tool altogether.**

☐**If the model tends to give up after many turns, I have tried including automatic reminders to keep trying and calling tools** (e.g., if the model hasn't called a tool in its last *n* turns)

Simple elicitation through prompting

☐I have exhaustively experimented with prompting techniques.

☐I have tried chain of thought reasoning (and I am recording the amount of reasoning tokens used).

☐If the model's chain of thought often goes off the rails immediately, I have tried *prefilling* or *strong hinting* to push its thinking in the right direction.

Prefilling involves prompting the model with a statement such as "I'm about to try method X." Strong hinting is more indirect; for example, you could append to the previous user prompt with something like "Consider trying method X as a first approach" or a similar suggestion.

☐ If task examples don't exist or are too large for the context window (e.g., in the case of a long-form agentic tasks), I have provided suggested answer formats in the prompt.

☐If task examples do exist for this task and fit in the context window, I have tried few-shot prompting.

☐If the model is often failing due to forgetting important parts of the prompt, I have experimented with multi-turn prompting.

Reporting

What does a good report look like?

A good report begins with a clear and intuitive explanation of the proposed elicitation method, ideally accompanied by a visual sketch. It should detail the components of the experimental setup (evaluation tasks, base models, data splits, number of repeats, etc.), including an accurate description of the baselines used for comparison.

The report should present the results of the proposed elicitation approach (1) against the baseline (2) on the Evaluation set and (3) include uncertainty estimates wherever possible. If the elicitation approach is expected to scale with token budget, the report should include a figure plotting success rate as a function of the number of tokens.

Additionally, an analysis focusing on error cases and instances of improved performance over the baseline should be provided; this can be done through manual

qualitative inspection or transcript analysis. The report should be well-organized and span 2-4 pages, plus an appendix. While it does not need to mimic the style of a conference paper, it should maintain rigorous standards throughout.

☐I compared the performance of the target elicitation setup against baselines on the Evaluation set.

☐I produced a report detailing the overall results of the evaluation and any major red flags raised.

☐I shared it with others to disseminate the results.