Project Purpose

Project Google Colab

Our project is about figuring out how much attention a language model gives to different parts of a conversation when it's answering a question. Imagine the model is like a person trying to listen to three voices at once: one voice gives it instructions (the "system prompt"), another asks a question (the "user"), and the third is its own thoughts as it starts talking (its "self-context"). We want to measure which voice it listens to the most , can it be the instructions, the question, or itself. We will test this while it's writing its answers.

The big question I think … is so what? So what if a model pays too much attention to its system prompt or the user, why do we care?

The main answer for why we care what the model listens to most is because it tells us how and why it behaves the way it does. If a model pays too much attention to its *system prompt*, it might always follow generic instructions ("be polite," "summarize," make the user feel like a genius, add 39 emojis to a text) and ignore what the user actually asked. This can be quite bad for usefulness. If it pays too much attention to *itself* (its prefix), it might start drifting off topic or hallucinating. But if it balances attention well between the *prompt* and the *user's question*, it can stay accurate and responsive.

By measuring where the model's attention really goes, we can understand whether instruction tuning (the process that makes models follow directions better) is helping or hurting that balance. It's a way of opening up the black box to see if models are actually listening to *us* or just to themselves.

To do this, we look inside the model's "attention," which is like a map showing where it's focusing at every moment. Using math, we can count how much attention goes to each part of the input and turn that into three easy-to-understand numbers. Then, we compare models that were trained on different kinds of data — for example, chatty conversations versus factual question-answer data — to see how their habits differ. This helps us understand how training changes the way models think and whether they're following the user or just repeating what they were told.

Project Schedule

Measuring Prompt Reliance Across Instruction-Tuning Datasets

Week 1 – Setup & Dataset Preparation

Goal: get everything running and gather clean data.
Tasks:

- Install and verify required tools (`TransformerLens`, `PyTorch`, `datasets(alpaca,flan,sharegpt)`, `transformers`).
- Select and preprocess ~300 examples from **Alpaca**, **FLAN**, and **ShareGPT** into *(system, user, assistant)* triplets.
- Transformers well be using are: LLaMA-2-7B (Meta AI), Mistral-7B (Mistral AI)
- Standardize tokenization and format for all datasets.
- Run a small sanity check with a toy model.

Summary of what needs to be done in week 1

- dataset selection, cleaning, and formatting.
- environment setup, dependency management, model download.
- documentation of setup process and data-source justification.

Deliverable: Ready-to-use dataset + reproducible setup instructions.

Week 2 – Attention Extraction Pipeline

Goal: capture the model's internal attention patterns.
Tasks:

- Use `TransformerLens` hooks to record post-softmax attention weights layer-by-layer.
- Separate tokens by segment (P = Prompt, U = User, A = Assistant).
- Compute and store attention matrices for a few test samples.
- Visualize sample heatmaps.

Roles

- **Benny:** implement hooks and extraction script.
- **Arsh:** verify correct segmentation of P/U/A tokens.
- **Avi:** test runs, debug logging, write mini-report on extraction process.

Deliverable: functioning extraction notebook + sample visualizations.

Week 3 – Metric Computation (PAM/QAM/SAM)

Goal: calculate quantitative reliance metrics.
Tasks:

- Write functions to compute normalized attention shares (PAM, QAM, SAM).
- Validate normalization (sum = 1).
- Run across several examples per dataset.
- Generate first summary plots (bar charts per dataset).

Roles

- **Benny:** code metric computation functions.
- **Arsh:** validate math consistency and sanity-check values.
- **Avi:** visualize and interpret early trends.

Deliverable: working metric pipeline + initial numerical results.

Week 4 – Causal Ablations and Baseline Models

Goal: test causal importance and baselines.
Tasks:

- Zero out P→A or U→A edges to compute Prompt Contribution to Logits (PCL).
- Compare instruction-tuned (Chat) vs. base models.
- Add randomized-prompt and shuffled-query baselines.

Roles

- **Benny:** implement ablation code and run experiments.
- **Arsh:** manage baseline datasets and control runs.
- **Avi:** analyze differences between base vs chat models; document findings.

Deliverable: PCL results + baseline comparison report.

Week 5 – Analysis and Visualization

Goal: interpret and present what the model “listens to.”
Tasks:

- Aggregate all results across datasets.
- Create comparative charts (PAM/QAM/SAM distributions per dataset).
- Correlate with behavior metrics (instruction adherence, BLEU/F1).
- Draft main results figures for final report.

Roles

- **Avi:** lead interpretation and writing of discussion section.
- **Benny:** build visualizations and ensure statistical clarity.
- **Arsh:** finalize dataset tables and cross-validation checks.

Deliverable: analyzed results + figures ready for inclusion.

Week 6 – Final Report and Presentation

Goal: synthesize everything into a coherent deliverable.
Tasks:

- Write the final paper (introduction, methods, results, discussion).
- Prepare slides or demo visualizations for presentation.
- Peer-review each other’s sections for clarity and formatting.

Roles

- **Avi:** compile and polish final report; coordinate submission.
- **Benny:** proofread methods/math and verify figures.
- **Arsh:** edit dataset and results sections; finalize references.

Deliverable: complete project report and presentation ready for submission.