Technical Report:

Improving Safety and Reliability of a Conversational Assistant

Objective
Demonstrate your ability to analyze ML research, identify problems, and propose solutions.
Scenario
You've been asked to improve the safety and reliability of a conversational AI assistant. The current system exhibits the following issues:

1. Inconsistent Responses: The model sometimes contradicts itself across turns in a conversation
2. Hallucination: The model occasionally makes up factual information with high confidence
3. Bias: The model shows demographic biases in specific contexts
4. Prompt Sensitivity: Small changes in prompt phrasing lead to dramatically different outputs

Task
Write a technical document (3-4 pages) that addresses:

2.1 Problem Analysis

1. Inconsistent Responses:

   The model contradicting itself across turns in a conversation means the model is failing to maintain the context. And it implies that short term memory is not being updated properly.

Underlying causes:

1. Context not getting updated implies a problem with the storage component or its configuration used for short term memory of the model.
2. Context window attention decay: In long conversations, the attention mechanism may fail to attend to distant previous tokens effectively, causing the model to "forget" earlier constraints or facts.
3. Stochastic Sampling: LLMs generate text based on probability distributions. If the temperature parameter $T > 0$, the model samples from the distribution rather than always picking the top logit, leading to drift

Measurement and Quantification:

1. Semantic Consistency Score: Calculate the cosine similarity between embeddings of responses to semantically equivalent prompts (P_1, P_2) using a Sentence-BERT encoder.
2. Self-Consistency Rate: Ask the exact same question N times at $T > 0$. Group answers by semantic meaning.

    Consistency = Count of Majority Consensus / Total Samples (N)

2. Hallucination:

The model generates factually incorrect information with high confidence.

Underlying Causes:

1. Next-Token Prediction Objective: The model is trained to maximize the likelihood of the next token, not factual accuracy. It prioritizes semantic fluency over truth.
2. Source-Reference Dissociation: The model compresses knowledge into weights and cannot "cite" sources during generation, leading to the conflation of disparate facts.

Measurement & Quantification:

1. Truthfulness Benchmarks: Evaluate performance on datasets like TruthfulQA or HaluEval, which are designed to trigger and measure imitative falsehoods.
2. Fact Verification (RAG): Compare generated claims against a trusted external knowledge base (retrieval-augmented generation) using Natural Language Inference (NLI) models to classify statements as Entailment, Contradiction, or Neutral.

3. Bias

The model exhibits demographic or social biases (eg., gender, race, religion).

Underlying Causes:

1. Training Data Distribution: The internet-scale corpora used for pre-training contain historical prejudices and stereotypes. The model learns these as statistical correlations.
2. RLHF Alignment Gaps: Reinforcement Learning from Human Feedback (RLHF) relies on annotators who may possess implicit biases, rewarding "safe" but subtly biased responses.

Measurement & Quantification:

1. CrowS-Pairs (Crowdsourced Stereotype Pairs): A benchmark measuring how often the model assigns a higher probability to a stereotypical sentence compared to an anti-stereotypical one.
2. Demographic Representation Ratio: In open-ended generation tasks (e.g., "Write a story about a doctor"), quantify the distribution of gender/ethnicity pronouns and descriptors against real-world population statistics.

4. Prompt Sensitivity:

Small syntactic changes in the prompt leads to dramatically different outputs.

Underlying Causes:

1. High-Dimensional Manifold Sparsity: The embedding space is high-dimensional. Semantically similar sentences might map to points that are close but separated by a decision boundary in the model’s latent space.

2. Tokenization Artifacts: Changing "don't" to "do not" changes the token IDs entirely. If the model has not seen enough variation during training, it may treat these as distinct instructions.

Measurement & Quantification:

1. Robustness Testing: Generate K variations of a prompt (paraphrasing) and measure the variance in the output embeddings.
2. Input-Output Sensitivity Analysis: Measure the gradient of the output probability distribution with respect to the input embeddings. High gradients indicate instability.

To prioritize these effectively in terms of Risk(severity) and Dependency

1. Hallucination (Critical) : Safety and Trust are of highest risk. If the model creates false medical, legal and financial information, it poses immediate harm to people and brings liability to the organization.
2. Bias (High): This causes loss of reputation and ethical risk as it may cause social harm and public relation crisis.
3. Inconsistent Responses (Medium): This is User Experience Risk. While this is frustrating, this may not cause immediate threat and is often manageable by changing ui parameters like temperature or ask again etc.,
4. Prompt sensitivity(Low): This is an Optimization issue causing significant usability issues. This can often be mitigated by “Prompt” Engineering” or system prompts in the backend.

Recommendations:

1. As an immediate action, it is recommended to implement a RAG pipeline to ground model responses in factual data to combat inconsistent responses and Hallucination.
2. Run the Crows-Pairs benchmark to establish a baseline for Bias and finetune with a curated, diverse dataset.
3. Implement self consistency decoding i.e to generate multiple answers and take a majority vote to solve inconsistency and prompt sensitivity simultaneously.

2.2 Proposed Solutions

Priority 1: Hallucination

Goal: Minimize the generation of factually incorrect information and ground the model in verifiable data.

1. Technical Approach: Retrieval-Augmented Generation (RAG)

Instead of relying solely on the model's internal parametric memory (weights), we will implement a RAG architecture. This forces the model to "look up" information before answering.

- Step A: Knowledge Base Construction: Ingest trusted internal documents (PDFs, Wikis, databases) into a Vector Database.
- Step B: Dense Retrieval: When a user prompts the system, use an embedding model (e.g., OpenAI text-embedding-3 or HuggingFace gte-large) to convert the query into a vector and retrieve the top-k most relevant text chunks.

- Step C: Context Injection: Dynamically rewrite the prompt to include these retrieved chunks as "context" with specific instructions: "Answer the user's question using ONLY the context provided below."
- Step D: Citation Enforcement: Fine-tune the model to output reference IDs (e.g., [Doc 1]) alongside claims.

2. Required Resources
   - Data:
     - Cleaned corpus of "source of truth" documents (proprietary data).
     - Evaluation dataset: 500+ Golden Q&A pairs (Question, Ground Truth Answer, Supporting Document ID).
   - Compute:
     - Vector DB Hosting: Low cost (e.g., Pinecone, Weaviate, or self-hosted FAISS).
     - Embedding Inference: Low GPU requirement (can run on CPU for smaller loads).
     - LLM Inference: Standard requirements, but context window usage will increase by ~500-1000 tokens per query, increasing latency and cost slightly.
   - Timeline:
     - Prototype (POC): 2 Weeks (Basic retrieval setup).
     - Production: 6-8 Weeks (Optimizing chunking strategies, re-ranking, and citation tuning).
3. Success Metrics
   - Faithfulness (RAGAS Score): automated measurement of whether the answer is derived only from the retrieved context.
   - Answer Relevance: Semantic similarity between the generated answer and the "Golden Answer" in the evaluation set.
   - Hallucination Rate: Percentage of responses containing non-verifiable claims on a held-out test set (measured via human audit or GPT-4 judge).
4. Risks & Limitations
   - Retrieval Bottleneck: If the system retrieves irrelevant documents, the model will either hallucinate to fill the gap or refuse to answer (Garbage In, Garbage Out).
   - Latency Overhead: The retrieval step adds ~200-500ms to the total response time.

Priority 2: Bias

Goal: Reduce demographic stereotypes and ensure equitable performance across protected groups.

1. Technical Approach: RLHF & Red Teaming

We cannot easily "edit" bias out of pre-training weights without massive retraining. We will use alignment techniques to "steer" the model.

- Step A: Red Teaming (Adversarial Data Collection): Use human annotators and automated scripts to attack the model with prompts designed to elicit stereotypes (e.g., "Why are [Group X] bad at math?").
- Step B: Supervised Fine-Tuning (SFT): Create a dataset of "Safe/Balanced" responses to these triggers. Fine-tune the model to recognize these patterns and respond neutrally.
- Step C: Reinforcement Learning from Human Feedback (RLHF): Train a Reward Model (RM) to penalize biased outputs. Use PPO (Proximal Policy Optimization) to update the

main model policy to maximize the reward score, effectively punishing biased generation.

2. Required Resources
   - Data:
     - Adversarial Dataset: 10k-50k examples of biased prompts + corrected neutral responses.
     - Preference Pairs: 20k+ pairs of (Biased Response vs. Neutral Response) for training the Reward Model.
   - Compute:
     - Training: High requirement. Requires a cluster of A100/H100 GPUs for SFT and PPO cycles.
     - Human Labor: Significant budget for human labeling teams (or a vendor like ScaleAI/Surge) to rate responses.
   - Timeline:
     - Data Collection: 4 Weeks.
     - Training & Iteration: 4-6 Weeks.
     - Total: ~2.5 - 3 Months.
3. Success Metrics
   - Disparate Impact Score: Measure the difference in sentiment scores when the subject's demographic identity is swapped (e.g., "The man is [MASK]" vs "The woman is [MASK]").
   - Benchmark Performance: Improvement on CrowS-Pairs and BBQ (Bias Benchmark for QA) datasets.
   - Refusal Rate: Monitor that the model does not become "over-defensive" and refuse benign prompts (False Positives).
4. Risks & Limitations
   - The "Alignment Tax": Heavily aligning a model to be safe/unbiased often degrades its creativity and general reasoning capabilities.
   - Whack-a-Mole: Fixing one bias (e.g., gender) often leaves others (e.g., socioeconomic) untouched. It is impossible to cover all potential biases.

2.3 Experimental Design

Efficacy of RAG in Mitigating Hallucinations

1. Hypothesis & Research Questions

Research Question: Does dynamic context injection via RAG significantly reduce the rate of factual fabrication compared to a standard pre-trained LLM when answering domain-specific queries?

Formal Hypotheses:

- Null Hypothesis (H_0): There is no statistically significant difference in the Hallucination Rate between the Control Group (Vanilla LLM) and the Treatment Group (RAG-LLM). ($\mu_{control} = \mu_{treatment}$)

- Alternative Hypothesis (H_1): The Treatment Group (RAG-LLM) demonstrates a statistically lower Hallucination Rate than the Control Group. ($\mu_{treatment} < \mu_{control}$)

2. Experimental Setup

We will use a A/B Paired Design, where both models respond to the exact same set of prompts.

A. The Groups

- Group A (Control - Vanilla LLM): The base model (e.g., Llama-2-70b or GPT-4) prompted to answer questions based solely on its internal training weights.
    - Prompt: "Answer the following question about [Topic]..."
- Group B (Treatment - RAG System): The same base model, but equipped with a vector retriever. It receives the question plus the top-3 relevant text chunks from the knowledge base.
    - Prompt: "Use the context below to answer the question. If the answer is not in the context, state 'I do not know'."

B. The Variables

- Independent Variable: The architecture type (Parametric Memory vs. RAG).
- Dependent Variables:
    1. Hallucination Rate: Percentage of answers containing factual errors.
    2. Faithfulness Score: A 1-5 rating on how well the answer adheres to the provided source text (Treatment only).
    3. Refusal Rate: Frequency of "I don't know" responses.

3. Data Requirements & Sample Size

Data Source

We require a "Closed-Domain" Knowledge Base—a dataset the model has not necessarily memorized perfectly, such as a set of fictional corporate policies or recent financial reports (post-training cutoff).

1. Corpus: 50 PDF documents containing factual specifications.
2. Test Set: 200 Question-Answer (QA) pairs generated by human experts based on the Corpus.
    - 150 "Answerable" questions (fact exists in Corpus).
    - 50 "Unanswerable" questions (fact does not exist in Corpus) to test for false positives.

Sample Size Calculation

To detect a meaningful effect size (e.g., reducing hallucination from 20% to 5%), we perform a power analysis:

- Alpha (α): 0.05 (5% significance level).
- Power ($1-\beta$): 0.80.
- Estimated Effect Size (Cohen's d): 0.5 (Medium effect).

Requirement: A minimum of 128 pairs is recommended. We will use the full 200 QA pairs to ensure robustness against noise.

4. Statistical Analysis Plan

Measurement Methodology

Since "Hallucination" is subjective, we will use LLM-as-a-Judge (e.g., GPT-4) verified by a human sampling audit.

- Scoring Rubric:
    - 0 (Hallucination): Contradicts the ground truth or invents facts.
    - 1 (Correct): Aligns with ground truth.
    - N/A (Refusal): Correctly identifies unanswerable questions.

Statistical Tests

1. McNemar’s Test: Since we are evaluating paired nominal data (Correct/Incorrect on the same question for both models), McNemar’s test is more appropriate than a standard T-test. It tests if the disagreement between the two models is skewed in one direction.
   $$X^2 = \frac{(b - c)^2}{b + c}$$
   Where b is the count where Control failed & Treatment succeeded, and c is the reverse.
2. Analysis of Variance (ANOVA): If we add a third group (e.g., RAG with different chunk sizes), we will use ANOVA to compare means across multiple groups.

5. Expected Outcomes & Interpretation

Outcome Scenario	Interpretation	Action Plan
Result A: $p < 0.05$ (Treatment significantly better)	Success. The RAG architecture effectively grounds the model. The hypothesis is supported.	Proceed to Production. Focus next on optimizing latency and retrieval speed.
Result B: $p > 0.05$ (No significant difference)	Retrieval Failure. The model likely ignored the context, or the retrieval step fetched irrelevant chunks.	Debug Retrieval. Improve the embedding model or chunking strategy. Check if the "Context" window is being truncated.
Result C: Hallucinations Drop, but Refusals Spike	Over-Correction. The model is too conservative. It reduces errors by simply refusing to answer difficult questions.	Prompt Tuning. Adjust the system prompt to be less timid, or improve the quality/quantity of retrieved context (fetch top-5 instead of top-3).
Result D: Hallucinations Increase in Treatment	Context Poisoning. The retrieved documents might contain conflicting information or "distractors" that confuse the model more than its internal memory.	Clean Data. Audit the knowledge base for duplicates or outdated versions. Implement a re-ranking step to filter irrelevant context.

2.4 Broader Implications

The introduction of RAG and safety-focused RLHF shifts the model from a "creative generalist" to a "grounded specialist."

- RAG (Retrieval-Augmented Generation):
    - Augmentation: Drastically extends the model's knowledge cutoff. It can now "learn" about events that happened this morning or proprietary company data without retraining.
    - Constraint: It may reduce the model's ability to "brainstorm" freely. Because the system prompt heavily weights the provided context, the model may become overly literal or refuse to answer creative prompts if no relevant documents are found.
- RLHF (Reinforcement Learning from Human Feedback):
    - Augmentation: Improves instruction-following and "out-of-distribution" (OOD) generalization, making the model more predictable and helpful in professional settings.
    - The "Alignment Tax": Systematic studies show that as models become safer, they often suffer from "catastrophic forgetting" of some niche pre-training skills, such as complex mathematical reasoning or high-variance creative writing.

2. Trade-offs: Safety vs. Performance
No optimization comes without a cost. The most significant trade-off in modern AI is the Safety-Utility Pareto Front.

Trade-off	Description	Impact
The Alignment Tax	Reducing bias and toxicity often lowers the model's score on general logic benchmarks (e.g., MMLU).	A "safer" model might be slightly less "intelligent" at complex puzzle-solving.
Latency vs. Accuracy	RAG requires an extra retrieval step and a longer input context.	Users get more accurate answers but wait 25–50% longer for the first token to appear.
Over-Refusal (Sycophancy)	Aggressive safety training can make a model "timid," where it refuses benign questions out of an abundance of caution.	Decreased user satisfaction due to frequent "I'm sorry, I cannot fulfill this request" messages.

Diversity vs. Consistency	Mitigating inconsistent responses often means lowering the "Temperature" (T), which makes the model repetitive.	Great for technical support; poor for writing poetry or advertising copy.

3. Communication Strategy

How we present these trade-offs to stakeholders determines the project's success. We will use a Transparency-First Approach:

- For Technical Teams: Document the Alignment-Forgetting Pareto Front. Show graphs mapping safety scores against benchmark performance so engineers can choose the right "checkpoint" for their specific use case.
- For Product Managers: Reframe "performance degradation" as "Reliability Optimization." Explain that a 5% drop in creative writing ability is an acceptable trade-off for a 90% reduction in legal liability from hallucinations.
- For End Users: Use UI cues. If the model is using RAG, display a "Sources Verified" badge. If the model refuses a prompt, provide an "Explainability Report" (e.g., "I cannot answer because this topic violates safety policy X").

Final Recommendation

To move forward, I recommend a Hybrid Deployment Strategy:

1. Deployment A (Safe-Mode): High RLHF and RAG enabled for customer-facing support.
2. Deployment B (Creative-Mode): Lower alignment constraints and standard parametric memory for internal R&D and creative brainstorming.