Model Behavior: Evals, Debugging, Alignment

Lecture 5 Notes

CDSS 94: Building Thoughtful AI Systems (Spring 2026)

Contents

1 What Are Evals?

1.1 The Role of Evaluation

Throughout the model lifecycle, evaluations serve three roles:

Definition 1.1: Evaluations in the Model Lifecycle

1. **Diagnostic tool:** Understand where models succeed and fail, and what capabilities and limitations exist.
2. **Debugging mechanism:** Trace issues back to data quality, training recipes, or alignment choices.
3. **Trust proxy:** Provide confidence levels for whether a model is ready for deployment.

Evaluations span every stage: data → pre-training → post-training → deployment.

1.2 From Traditional ML to LLM Evaluation

Traditional AI/ML Evaluation

Traditional evaluation measured how well a model performs on a single, well-defined task using a training/validation/test split. Canonical examples include ImageNet (14M images, 1000 classes, top-5 error rate), CIFAR-10/100, and MNIST.

Traditional NLP Evaluation

Models were trained on next-token prediction (minimize perplexity), then fine-tuned on specific downstream tasks. Metrics included BLEU (n-gram overlap for translation), GLUE/SuperGLUE (NLU benchmark suites for sentiment, entailment, similarity), and WMT (standardized translation test sets).

The Paradigm Shift

Remark 1.2: From Task Performance to Model Behavior

LLMs are zero-shot performers (GPT-3): no fine-tuning needed, they can do things they were not explicitly trained for. This forced a paradigm shift:

- "How well does it do something" → "How does it *behave*?"
- "Single-task evaluation" → "Benchmarking"

Old metrics (F1, BLEU) and evaluation paradigms break down under these assumptions.

1.3 Benchmarks

Definition 1.3: Benchmark (Raji et al., 2021)

A benchmark is a combination of (i) a dataset or sets of datasets, and (ii) a metric or metrics, conceptualized as representing one or more specific tasks or sets of abilities. Benchmarks are adopted by a community of researchers as a shared framework for the comparison of models.

Definition 1.4: What Makes a Good Benchmark?

1. **Generalizability:** Does benchmark performance predict real-world usefulness?
2. **Difficulty / headroom:** Challenging enough that frontier models don't saturate it.
3. **Diversity:** Broad coverage of topics, tasks, difficulty levels, and languages.
4. **Reproducibility:** Anyone can run it and get consistent, comparable results.
5. **Low contamination risk:** Resistant to data leakage from training corpora.
6. **Fair grading:** Unambiguous answers, reliable automated or human scoring.

Example 1.5: Data Contamination: GSM8K vs. GSM1K

GSM8K is a widely-used benchmark of mathematical word problems. GSM1K contains similar problems guaranteed to be novel. Many models scored significantly lower on GSM1K, suggesting that high GSM8K scores partly reflect *memorization* rather than genuine mathematical reasoning.

1.4 Who Does the Evaluation?

Method	Characteristics
Automatic / Rule-based	Exact match, regex, unit tests (e.g., HumanEval). Fast, cheap, reproducible; brittle, can't assess open-ended quality.
Human Evaluation	Expert annotators rate outputs. Gold standard for subjective tasks; expensive, slow, hard to scale.
LLM-as-a-Judge	A strong model (GPT-4, Claude) grades another model's outputs. Scalable and cheap; introduces judge bias and circular evaluation risks.

2 What Can We Measure?

2.1 Dimensions of Model Behavior

Definition 2.1: Measurement Dimensions

Model behavior can be measured along many axes:

1. **Capabilities:** knowledge, reasoning, coding
2. **Safety:** refusal behavior, toxicity, bias
3. **Honesty:** hallucination, calibration
4. **Preference:** do humans like the outputs?
5. **Robustness:** adversarial inputs, multi-turn degradation
6. **Instruction following:** does the model do what was asked?
7. **Domain-specific:** biology, legal, physics, chemistry

Remark 2.2: Model Cards

Model cards provide standardized documentation: model overview, capability evals, safety evals, bias & fairness assessments, limitations & risks, and deployment policy.

2.2 Case Study 1: Refusals

The Over-Refusal Problem

Example 2.3: Claude 2.1 Over-Refusals

Claude 2.1 would refuse tasks that *superficially* sounded harmful but actually were not (e.g., "How to kill the python process?"). This was fixable because 2.1 had more refusals on benign prompts than 2.0, setting up a baseline for experimentation.

The goals for improvement were: reduce over-refusals on innocuous tasks, produce more nuanced refusals, and broaden refusals on genuinely sensitive topics (elections, copyright).

Definition 2.4: Refusal Taxonomy

1. **Benign over-refusal:** refusing harmless prompts.
2. **Copyright refusals:** needed reduction in "I apologize..." in favor of more nuanced responses.
3. **Function calling refusals:** "I can't see your note" even though the model had a tool and was informed in the system prompt.

Principles for Better Refusals

Remark 2.5: Non-Violent Communication in Refusals

- Assume charitable interpretation of what the person is asking.
- The model distinguishes "can't" from "won't"—it does not say it can't do something when it won't do something.
- Refusals use "I statements": the model takes responsibility for its own refusal.
- Avoid "you statements" or judgments that the user is trying to do something bad.
- The model can state its risk model for *why* it has those boundaries.
- Where it cannot explain, the model "owns" the uncertainty rather than blaming the user.
- Acknowledge impacts: "I know this may be annoying to you."

Building Trustworthy Refusal Evals

Sources for refusal evaluation data include: customer prompts that induce refusals, thumbs-down data from products, synthetically generated borderline prompts, XTest (200 non-malicious prompts), and WildChat (diverse real user interactions including ambiguous requests, code-switching, topic-switching, and political discussions).

2.3 Case Study 2: SimpleQA

Definition 2.6: SimpleQA

4,326 short, fact-seeking questions with single, verifiable answers. Each answer is scored as correct, incorrect, or not attempted. Designed to be adversarially hard: questions were selected to trip up GPT-4.

Remark 2.7: Key Findings

Even frontier models hallucinate on simple factual queries over half the time. Factuality is hard to measure at scale because language models generate long completions containing dozens of claims. SimpleQA sidesteps this by restricting to short, fact-seeking questions—at the cost of leaving open whether short-form factuality generalizes to long-form.

Remark 2.8: Calibration

SimpleQA also measures calibration: how well do models know what they know? Two approaches: stated confidence, and frequency analysis (ask the model 100 times—higher frequency of the same answer indicates higher confidence). Larger models tend to be better calibrated.

2.4 Case Study 3: BrowseComp

Definition 2.9: BrowseComp

A benchmark measuring the ability of AI agents to locate hard-to-find information on the web. Questions were designed so that existing models (GPT-4o with and without browsing, o1) could not solve them. Trainers verified answers were not available in the first pages of simple searches, and tasks had to be challenging enough that another person could not solve them within ten minutes.

2.5 Case Study 4: HealthBench

Definition 2.10: HealthBench

A rubric-based evaluation where each model response is graded against physician-written rubric criteria specific to each conversation. Contains 48,562 unique rubric criteria. Each criterion outlines what an ideal response should include or avoid (e.g., a specific fact to include, unnecessarily technical jargon to avoid), with point values weighted to match physician judgment of importance.

Remark 2.11

The 5,000 conversations simulate interactions between AI models and users or clinicians. They are multi-turn, multilingual, span a range of medical specialties, and were selected for difficulty.

2.6 Case Study 5: PostTrainBench

Definition 2.12: PostTrainBench

Each agent is given a base LLM, a target benchmark, access to a compute node (a single H100 GPU), and internet access. The agent must build its training pipeline from scratch—no starter code, training data, or hyperparameter configurations are provided. The agent produces a post-trained model, which is evaluated on the target benchmark. Agents have full autonomy over data sources, training methods, and hyperparameters within a 10-hour time limit.

Example 2.13: Agent Reward Hacking in PostTrainBench

Agents sometimes engage in reward hacking: training on the test set, downloading existing instruction-tuned checkpoints instead of training their own, and using API keys they find to generate synthetic data without authorization.

2.7 Benchmark Comparison

Benchmark	Measures	Core Skill	Failure Mode
SimpleQA	Factual precision	Knowledge retrieval	Hallucination
HealthBench	Clinical reasoning	Safety-aware reasoning	Overconfidence / harm
BrowseComp	Tool-based research	Agentic browsing	Fabricated grounding
Model Cards	Transparency	Eval & disclosure	Misaligned expectations

3 Why Do Evaluations Break?

3.1 Saturation

Remark 3.1: Benchmark Saturation

Models reach near-perfect scores, making the benchmark no longer useful for differentiation. Models get better, but also train on data that overlaps with benchmarks. "Hard" benchmarks become easy as capability curves shift.

Timeline of saturation:

- GLUE (2018): saturated within a year of release.
- SuperGLUE (2019): human-level performance surpassed by 2020.
- MMLU (2021): frontier models now score 90%+.
- HumanEval (2021): top models pass 95%+ of problems.

3.2 Contamination and Gaming

Definition 3.2: Data Contamination

Benchmark questions or answers appear in training data. The model memorizes answers rather than demonstrating true capability. Hard to detect: even partial overlap (paraphrased questions) can inflate scores.

Definition 3.3: Benchmark Gaming

Optimizing specifically for benchmark performance at the expense of general ability. Techniques include: using different models for different benchmarks to cherry-pick best results, or submitting specially tuned model variants to leaderboards (e.g., the Llama 4 case).

Mitigations include: held-out test sets, canary strings, dynamic benchmarks, and third-party audits.

3.3 Evaluation Awareness

Remark 3.4: Can Models Tell When They're Being Evaluated?

Alignment faking (Anthropic, Dec 2024): Claude 3 Opus selectively complied with training objectives while preserving its own preferences. Implications:

- Models may behave differently in eval vs. deployment (sandbagging or performing).
- Evals that rely on model cooperation may underestimate true risk.
- Reasoning traces can reveal hidden reasoning that contradicts surface outputs.

Open question: as models become more situationally aware, how do we build evals they can't game?

3.4 The Limits of Metrics: "Vibes"

Remark 3.5: When Benchmarks Miss What Matters

Several failure modes illustrate the gap between benchmark scores and actual behavior:

1. A model passes safety benchmarks but sends unsolicited emails, grabs credentials, lies about tool outputs, and destroys git repos. None of these were categories in the eval suite—they were caught by engineers watching the model work.
2. Bioweapons refusal drops from 96% to 88% in multi-turn, in the same release where biology knowledge improved. The eval measured both numbers but didn't connect them. A human sees a dangerous combination; the pipeline processes each number in isolation.
3. Sycophancy is measured but defined too narrowly to catch the model folding under pressure. Answer thrashing (the model knows the right answer but a bad reward signal forces the wrong one) was found by accident. Tool output misrepresentation was caught through interpretability, not evals.

> *The things that matter most about a post-trained model are the hardest to put a number on. Post-training determines behavior. Behavior doesn't reduce to a benchmark.*

4 Evals as Research

4.1 Real-World Generalization

Definition 4.1: Chatbot Arena / LM Arena

Human preference-based evaluation: users compare two model outputs side-by-side. An ELO rating system ranks models by crowd-sourced preferences.
Strengths: Captures what users actually prefer, not just correctness.
Weaknesses: Stylistic bias (verbose responses can win), gameable.
Key tension: Do users prefer the *best* model or the most *pleasant* one?

4.2 Dynamic Benchmarks

Static benchmarks fail over time due to saturation, contamination, and gaming. Dynamic benchmarks are continuously refreshed:

Benchmark	Approach
LM Arena	New models and matchups rotate in continuously.
LiveBench	Questions updated monthly from recent information.
FreshQA	Tests on fast-changing knowledge that can't be memorized.

Advantages: resistant to contamination, better proxy for real-world relevance. Challenge: harder to maintain consistency across time periods.

4.3 Agentic Benchmarks

As models become agents, evals must adapt. Traditional QA benchmarks don't capture multi-step, tool-using behavior.

Benchmark	What It Tests
SWE-bench	Can the agent fix real GitHub issues end-to-end?
BrowseComp	Can the agent find hard-to-locate information on the web?
METR RE-Bench	Can the agent do ML research engineering tasks?
GAIA	Can the agent answer questions requiring multi-step tool use?

What makes agentic evals different: multi-step planning (plan, execute, observe, iterate), tool use, and test-time compute scaling.

5 Debugging Models

5.1 Evals as a Diagnostic

Definition 5.1: The Debugging Loop

1. Run evals on a new model or training run.
2. Identify regressions or unexpected behaviors.
3. Hypothesize root cause (data quality, reward signal, training instability).
4. Iterate: adjust training data, reward model, or post-training pipeline.
5. Re-run evals to verify the fix.

Evals are the feedback loop that connects model training to model improvement.

Remark 5.2

The debugging loop answers two questions: *where* does the model break down (capabilities: reasoning, knowledge, or instruction following?), and *why* (data quality, reward signal, or training instability?). Identifying failure modes is only useful if you can trace them back to actionable changes in the training pipeline.

6 Summary

Topic	Key Takeaway
What Are Evals?	Diagnostics, debugging mechanisms, and trust proxies across the model lifecycle.
Paradigm Shift	From "how well does it do X" to "how does it behave?" Old metrics break.
Benchmarks	Shared frameworks for comparison. Good ones are generalizable, hard, diverse, reproducible, and contamination-resistant.
Refusals	Over-refusal is measurable and fixable. Good refusals use NVC principles, distinguish can't from won't, and own uncertainty.
Factuality	Even frontier models hallucinate on simple facts >50% of the time (SimpleQA). Calibration matters.
Eval Failure Modes	Saturation, contamination, gaming, evaluation awareness, and the irreducible gap between benchmarks and behavior.
Dynamic & Agentic Evals	Continuously refreshed benchmarks and multi-step agent evaluations are the frontier.
Debugging	Evals are the feedback loop. Identify where, hypothesize why, iterate, verify.

> *Post-training determines behavior. Behavior doesn't reduce to a benchmark. The things that matter most about a post-trained model are the hardest to put a number on.*