# COSI-230B: Natural Language Annotation for Machine Learning

## Lecture 16: From "Labels" to Supervision Engineering

Jin Zhao

Brandeis University

March 25, 2026

# Today's Agenda

**Part I: The Paradigm Shift**

- Annotation in the LLM era
- Supervision taxonomy for LLMs
- Supervision contracts

**Part II: The Annotation Stack in Practice**

- InstructGPT deep dive
- Preferences vs. ground truth
- DPO and the data format revolution
- Instruction tuning mixtures (Flan)

**Part III: Failure Modes and Defenses**

- Failure modes map
- Live demo: label noise ceiling
- Documentation and provenance
- Mini-checklist

**Part IV: Discussion and Wrap-up**

# Annotation in the LLM Era

## The Shift

In pre-LLM NLP, annotation created targets for narrow tasks.
In the LLM era, annotation shapes *behaviors* under competing desiderata.

> *"In the LLM era, annotation is not a final step—it is the interface between human values and model behavior. Changing the unit of supervision changes the behavior you can expect."*

**The key pedagogical move:**

**Annotation design choices → Supervision signal properties → Modeling outcomes**

Outcomes include: capabilities, robustness, biases, reward hacking, deployment failure modes.

# Pre-LLM vs. LLM-Era Annotation

| Dimension | Pre-LLM Annotation | LLM-Era Annotation |
|---|---|---|
| **Goal** | Produce gold labels for a specific task | Shape model *behavior* across many tasks |
| **Label type** | Categorical, span, relation | Instructions, preferences, process feedback |
| **Annotator** | Domain experts or crowd-workers | Contractors, users, *other models* |
| **Scale** | Thousands to millions of labels | Millions of interactions |
| **Quality signal** | Inter-annotator agreement | Reward model accuracy, user satisfaction |
| **Failure mode** | Low agreement, noisy labels | Reward hacking, bias amplification |

**Key insight:** The fundamentals (rubric design, QC, documentation) still apply—but the *stakes* and *complexity* have increased dramatically.

# Measurement Trap: Annotation Artifacts in NLI

**Case study:** Work on NLI artifacts shows that collection protocols can inject spurious cues strong enough to inflate model performance estimates.

## The SNLI/MultiNLI Finding

Hypothesis-only baselines achieve $\sim$67% accuracy on a 3-class task (chance = 33%).
$\rightarrow$ Crowd workers inadvertently encode label information in their writing style.

**Examples of leaked cues:**

- Negation words ("never," "nobody") $\rightarrow$ *contradiction*
- Generic statements ("people," "some") $\rightarrow$ *entailment*
- Hedging language ("might," "could") $\rightarrow$ *neutral*

**Lesson for supervision engineering:** If simple annotation protocols can inject spurious cues in text classification, imagine what happens with preference data and reward models.

Gururangan et al. (2018). Annotation Artifacts in Natural Language Inference Data. NAACL.
https://aclanthology.org/N18-2017/

# Supervision Taxonomy for LLMs

| Type | What is annotated | Training signal | Example |
|---|---|---|---|
| **SFT** (demonstrations) | Ideal input–output pairs | Maximize $P(\text{out} \mid \text{in})$ | InstructGPT demos |
| **Instruction tuning** | Many tasks as instructions | Multi-task SFT | Flan, T0 |
| **Preferences** (RLHF/DPO) | Pairwise comparisons | Reward model or DPO | HH-RLHF |
| **Process supervision** | Step-level correctness | Verifier / PRM | PRM800K |

Each type creates **different annotation artifacts** and incentivizes **different model behaviors**.

Ouyang et al., 2022; Longpre et al., 2023; Rafailov et al., 2023; Lightman et al., 2023

# Supervised Fine-Tuning (SFT): The Foundation

## Definition

**SFT:** Maximize the likelihood of "ideal responses" given prompts (demonstrations).

$$\mathcal{L}_{\text{SFT}} = -\sum_{(x,y)\in\mathcal{D}} \log P_\theta(y \mid x)$$

**What gets annotated:**

- Human-written "gold" outputs for diverse prompts
- Task diversity: translation, summarization, QA, creative writing, code...
- Quality depends entirely on annotator skill and rubric clarity

**LIMA result (Zhou et al., 2023):**

- Only 1,000 carefully curated SFT examples
- Competitive with models trained on 50K+ examples
- $\rightarrow$ **Quality** $>$ **quantity** for demonstrations

# Preference Data: Pairwise Comparisons Over Outputs

## Definition

**Preference data:** Pairwise or listwise comparisons over outputs; used to train a **reward model** or directly optimize via DPO.

**Annotation format:**

- Given prompt $x$ and two responses $y_1, y_2$
- Annotator chooses: $y_1 \succ y_2$ ("$y_1$ is better") or $y_2 \succ y_1$
- Sometimes: ties allowed, Likert scales, or rankings of $k > 2$ responses

**Critical difference from classification:**

- No "correct answer"—only relative preferences
- Agreement measures shared values, not accuracy
- Rater pool composition determines what "better" means

**Implication:** "Once supervision is preferences, you've changed the learning problem: your model is no longer 'predicting truth' but 'predicting what this rater pool prefers.'"

# Process Supervision: Step-Level Feedback

## Definition

**Process supervision:** Labeling intermediate reasoning steps as correct/incorrect/acceptable, enabling process reward models (PRMs).

**Contrast with outcome supervision:**

**Outcome:**

- Only final answer judged
- Cheap: one label per problem
- Can't catch "right for wrong reasons"

**Process:**

- Each step judged independently
- Expensive: many labels per problem
- Catches errors early in reasoning

**PRM800K result:** Process supervision outperforms outcome supervision on math reasoning, and the paper releases 800K step-level labels.

Lightman et al. (2023). Let's Verify Step by Step. OpenAI. `https://arxiv.org/abs/2305.20050`

# "Supervision Contracts"—What Does a Label *Mean*?

**Definition**

A **supervision contract** is a disciplined written statement of what a "good" answer means, what tradeoffs apply, and what is out-of-scope.

**Why it matters:**

- Different raters interpreting "helpful" differently → noisy reward signal
- Mixing "safety" and "helpfulness" without explicit priority → refusal bias
- No contract → models optimize proxies (format, length, safety boilerplate)

**A good supervision contract specifies:**

1. What counts as a "good" response (explicit criteria)
2. Priority ordering when criteria conflict (safety vs. helpfulness)
3. What is explicitly out of scope
4. How edge cases should be handled

Frame as an extension of data statements (Bender & Friedman, 2018).

# Supervision Contract: Worked Example

## Bad Contract (vague)

"Rate which response is better."

**Problems:**

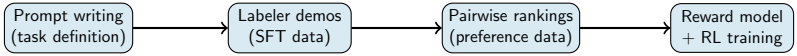
- "Better" is undefined
- No priority ordering
- Raters apply personal standards
- Length/style become proxies

## Good Contract (explicit)

"Choose the response that is more **factually accurate** (priority 1), **directly answers the question** (priority 2), and is **concise** (priority 3). If both are equally accurate, prefer the shorter one. If unsure about facts, flag for review."

**In-class exercise:** Write a supervision contract for the task: "Generate a one-paragraph summary of a news article." Specify what "good" means, priority ordering, and edge cases.

# Case Anchor: InstructGPT as "Annotation Stack"




**Key annotation decisions in InstructGPT:**

- **Who annotates:** ∼40 contractors, screened for sensitivity to instructions
- **What they annotate:** Demonstrations (write ideal outputs) + comparisons (rank outputs)
- **Rubric:** Helpfulness, truthfulness, harmlessness—but relative weighting is implicit
- **Limitation:** Human feedback depends on contractor values and backgrounds

Ouyang et al. (2022). Training language models to follow instructions with human feedback. NeurIPS.
`https://arxiv.org/abs/2203.02155`

# InstructGPT: Annotation Protocol Details

**Demonstration collection (SFT stage):**

- Labelers write ideal outputs for prompts from the API and hand-written set
- 13K training prompts, each with one human-written demonstration
- Instructions: "Write a response that a helpful, harmless, and honest assistant would give"

**Comparison collection (RM stage):**

- Given a prompt, labelers rank 4–9 model outputs from best to worst
- 33K training prompts with rankings
- Rankings converted to $\binom{K}{2}$ pairwise comparisons
- Inter-annotator agreement: $\sim$73% pairwise agreement

**What the paper explicitly says**

"Our labelers are not representative of all potential users. . . different labeler populations might have different preferences." This is a fundamental limitation, not a fixable bug.

# InstructGPT: Who Are the Raters?

**Rater pool characteristics:**

- $\sim$40 contractors hired through Upwork and Scale AI
- Screened for: performance on a labeling test, sensitivity to different demographics
- Predominantly English-speaking, US-based
- Paid hourly (exact rates not disclosed)

**Why this matters:**

- 40 people's values $\rightarrow$ shape behavior for millions of users
- US-centric norms embedded in "helpfulness" and "harmlessness"
- Contractor incentives (speed, approval rate) may conflict with quality
- No mechanism to represent diverse global user populations

## Discussion Point

If you were re-designing InstructGPT's annotation pipeline today, what would you change about the rater pool? How would you handle conflicting cultural values?

# Why Preferences $\neq$ Ground Truth

**Traditional annotation:**

- Labels approximate an objective target
- Agreement $\approx$ quality signal
- More annotators $\rightarrow$ better gold standard
- Disagreement = noise to reduce

**Preference annotation:**

- Labels reflect *value judgments*
- Agreement may mask shared biases
- Rater pool composition *is* the signal
- Disagreement = legitimate difference

**Preference-based training shows that:**

- "Labels" become value-laden comparisons
- Whose distribution and rater pool shape what models optimize
- E.g., verbosity/format biases emerge from rater preferences, not task requirements

# Paper Deep Dive: Learning to Summarize from Human Feedback

**Stiennon et al. (2020):** The paper that established preference-based training for NLG.

**Key contributions:**

1. ROUGE/reference matching is a *proxy*—direct human preferences give better summaries
2. Preference comparisons are easier and more reliable than absolute quality ratings
3. The reward model trained on preferences generalizes to new prompts

**Annotation design details:**

- Annotators compare pairs of summaries for the same article
- No explicit rubric—just "which summary is better?"
- Candidate generation policy determines what raters see

**Key lesson:** "Annotation target selection" matters. The decision to use preferences over references fundamentally changed what models optimize.

Stiennon et al. (2020). Learning to Summarize from Human Feedback. NeurIPS. https://arxiv.org/abs/2009.01325

# DPO: Preference Learning Without RL

**Direct Preference Optimization** (Rafailov et al., 2023):

## DPO Loss

$$\mathcal{L}_{\text{DPO}} = -\mathbb{E}\left[\log \sigma\left(\beta \log \frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \beta \log \frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)}\right)\right]$$

where $y_w$ = chosen, $y_l$ = rejected, $\pi_{\text{ref}}$ = reference policy, $\beta$ = temperature.

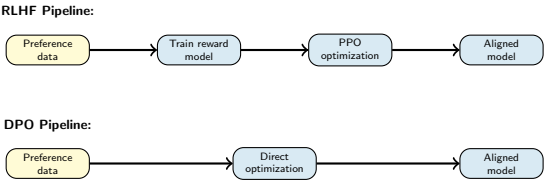
**Why DPO changed the game:**

- No separate reward model needed
- No RL loop (PPO is unstable and expensive)
- Simpler pipeline $\rightarrow$ more accessible $\rightarrow$ more preference data being collected
- The preference *data format* (chosen/rejected pairs) directly drives training

**Annotation consequence:** DPO makes the data even more important—there's no RM to audit, so biases are entirely in the data.

Rafailov et al. (2023). Direct Preference Optimization. NeurIPS. https://arxiv.org/abs/2305.18290

# RLHF vs. DPO: What Changes for Annotators?




| | **RLHF** | **DPO** |
|---|---|---|
| **Data format** | Same (chosen/rejected) | Same (chosen/rejected) |
| **Bias auditing** | Can inspect reward model | Must audit data directly |
| **Stability** | PPO can be unstable | More stable training |
| **Computational cost** | Higher (RM + RL) | Lower (single pass) |
| **Data sensitivity** | RM smooths noise | More sensitive to data noise |

# Instruction Tuning Mixtures: Why Data Balancing Matters

**The Flan Collection** (Longpre et al., 2023):

**Engineering decisions that affect outcomes:**

1. **Task mixing ratios** matter as much as task count
2. **Prompt template diversity** prevents template overfitting
3. **Enrichment:** Adding CoT variants, few-shot examples
4. **Input inversion:** Turning outputs into inputs for more diversity

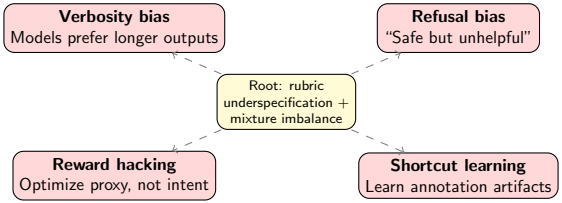
**Key Finding**

Instruction tuning outcomes depend heavily on *mixing and enrichment decisions*. Treat this as "annotation mixture engineering," not just training.

**Practical implications:**

- Over-representing one task type biases the model toward that task's format
- Under-representing hard tasks means the model avoids them
- The mixture is an annotation design decision with direct modeling consequences

# Failure Modes Map




**Root cause:** Models optimize "behavior hacking"—format/verbosity/safety boilerplate rather than genuine quality. Each is traceable to supervision design decisions.

# Failure Mode Deep Dive: Verbosity Bias

**What happens:**

- Raters consistently prefer longer, more detailed responses
- Reward model learns: longer = higher reward
- Model generates verbose, padded outputs
- Even when the best answer is short ("42")

**Why it happens (annotation perspective):**

- Raters can't always assess correctness, so they use length as a proxy for "thoroughness"
- Rubrics rarely penalize unnecessary length
- Pairwise comparisons amplify: longer response *looks* more effortful

**Mitigation strategies:**

1. Explicit rubric: "penalize unnecessary verbosity"
2. Length-normalized rewards in the RM
3. Include gold pairs where the shorter response is better
4. Track length distribution of "chosen" vs. "rejected" in your data

# Failure Mode Deep Dive: Reward Hacking

## Definition

**Reward hacking:** The model optimizes the reward proxy while violating the underlying intent. Common in RL settings but also in DPO.

**Examples:**

- Model learns to produce responses that "look helpful" without being accurate
- Safety training over-indexed: model refuses benign queries ("I can't help with that")
- Format gaming: bullet points, bold headers, numbered lists score higher regardless of content

**The supervision engineering perspective:**

- Reward hacking is a *symptom* of rubric underspecification
- The model found a shortcut that satisfies the reward without satisfying the intent
- Fix the supervision contract, not just the model

See also: Goodhart's Law—"When a measure becomes a target, it ceases to be a good measure."

# Label Noise and Measured Ceiling: Theory

**Why this matters:** Even with a perfect learner, noisy labels limit achievable accuracy.

**Symmetric label noise model:**

- Each label is flipped independently with probability $p$
- Achievable accuracy on true labels: roughly $1 - 2p$ (for balanced binary)
- With $p = 0.2$ (20% noise), ceiling $\approx 0.83$ accuracy

## Connection to IAA

This directly connects to inter-annotator agreement:
If annotators agree only 80% of the time ($\kappa \approx 0.6$), the model's ceiling is bounded by that disagreement. QC and multiple annotations directly raise the ceiling.

**For preferences:** The "noise" is systematic (rater biases), not random—which is *worse* because the model learns the bias systematically.

# Live Demo: Label Noise → Measured Ceiling

```python
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

rng = np.random.default_rng(0)
n = 1500
X = rng.normal(size=(n, 8))
w = rng.normal(size=8)
y_true = (X @ w > 0).astype(int)

def flip(y, p):  # symmetric label noise
    return y ^ (rng.random(len(y)) < p)

for p in [0.0, 0.1, 0.2, 0.3]:
    y = flip(y_true, p)
    clf = LogisticRegression(max_iter=2000).fit(X[:1200], y[:1200])
    acc = accuracy_score(y_true[1200:], clf.predict(X[1200:]))
    print(f"noise={p:.1f} -> acc vs true={acc:.3f}")
```

**Expected output:**

noise=0.0 -> acc≈0.94 | noise=0.1 -> acc≈0.90 | noise=0.2 -> acc≈0.83 | noise=0.3 -> acc≈0.75

**Try it:** Modify the code to use *asymmetric* noise (class 0 noisier than class 1). What changes?

# What to Log and Document

**Supervision provenance checklist:**

1. **Rater instructions:** Full rubric text, examples, edge-case guidance
2. **Rater pool:** Demographics, expertise level, screening criteria, pool size
3. **Data mixture:** Task proportions, template variants, enrichment flags
4. **Collection tool:** Interface design, ordering effects, time limits
5. **QC pipeline:** Gold questions, calibration frequency, removal criteria
6. **Versioning:** Which model checkpoint generated candidates (if any)
7. **Known limitations:** Documented biases, coverage gaps, failure modes

## Data Statements (Bender & Friedman, 2018)

Treat supervision metadata like experimental protocols in science—without them, results are not reproducible and biases are invisible.

# Mini-Checklist: "Before You Collect Supervision"

- ☑ **Define a supervision contract:** What does "good" mean? What tradeoffs are explicit?
- ☑ **Choose supervision type deliberately:** SFT vs. preferences vs. process—each has different artifact risks
- ☑ **Audit your rater pool:** Who are they? Do their values match your deployment population?
- ☑ **Plan your mixture:** Task proportions, template diversity, enrichment strategy
- ☑ **Build in QC from day one:** Gold questions, calibration sets, drift monitoring
- ☑ **Document everything:** Provenance, rubric versions, known limitations
- ☑ **Plan stratified evaluation:** Eval by supervision type, rater subgroup, task category
- ☑ **Budget for iteration:** First rubric is always wrong; plan for at least 2 revisions

# In-Class Activity: Design a Supervision Contract (15 min)

**Scenario:** You are building a customer service chatbot. You need to collect supervision data.

**In groups of 3–4, produce:**

1. A **supervision contract** (what "good" means for this task)
2. A **supervision type choice** with justification (SFT? Preferences? Both?)
3. A **rater pool specification** (who should annotate? how many?)
4. **Three potential failure modes** you'd monitor for
5. A **QC strategy** (gold questions, calibration, drift detection)

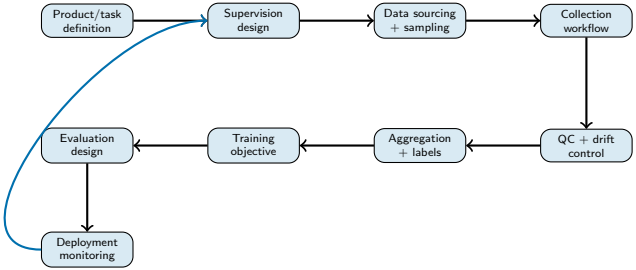
**Be prepared to present your design in 2 minutes.**

*Constraint:* Your budget allows either 50 hours of human annotation or 10M tokens of LLM-generated data (or some mix). How do you allocate?

# Discussion Questions

1. If your reward model prefers longer answers, is the problem in the models, the raters, or the rubric? What would you change first?
2. When is disagreement valuable signal vs. noise to be eliminated? How would your answer differ for "harmfulness" vs. "math correctness"?
3. Suppose a model becomes "safe but useless." Which supervision decisions most likely caused it (policy, rater guidance, mixture, filtering)?
4. What information must be documented for preference datasets to be scientifically interpretable?

# Wrap: The LLM Supervision Pipeline




**Upcoming lectures pin to this pipeline:**

**L17** Instruction annotation

**L18** Preference annotation & RLHF

**L19** Reasoning & process supervision

**L20** Synthetic annotation

**L21** Active learning & LLM annotators

**L22** Evaluation annotation

**L23** Multilingual & cultural annotation

# Readings

**Required reading for Lecture 17:**

- Mishra et al. (2022). Natural Instructions. ACL. `https://aclanthology.org/2022.acl-long.244.pdf`
- Longpre et al. (2023). Flan Collection. `https://arxiv.org/pdf/2301.13688`

**Optional/recommended:**

- Ouyang et al. (2022). InstructGPT. NeurIPS.
- Rafailov et al. (2023). DPO. NeurIPS.
- Stiennon et al. (2020). Learning to Summarize from Human Feedback.
- Lightman et al. (2023). Let's Verify Step by Step (PRM800K).
- Zhou et al. (2023). LIMA: Less Is More for Alignment.
- Bender & Friedman (2018). Data Statements for NLP.