COSI-230B: Natural Language Annotation for Machine Learning

Lecture 11: Annotation Agreement in the LLM Era

Jin Zhao

Brandeis University

March 2, 2026

Today's Agenda

1. Warm-up: What do κ and α assume?
2. What changes when the annotator is an LLM
3. Live demo: LLM as annotator (structured outputs)
4. Hands-on: Human–LLM and LLM–LLM agreement
5. Exit check

The Key Conceptual Pivot

After Lectures 8–10, you know how to compute κ and α.

Now the world has shifted:

Central idea for today

LLM annotation is **measurement with a moving instrument**.

"Inter-annotator agreement" becomes
"inter-annotator-and-configuration agreement."

Warm-up: What Do κ/α Assume?

Recall the assumptions behind classic chance-corrected coefficients:

1. **Independence** — annotators label items without consulting each other
2. **Stable marginals** — each annotator has a fixed tendency to use each label
3. **Hard labels** — each annotator produces exactly one categorical label per item

Question: Which of these still hold when one "annotator" is an LLM?

Warm-up Exercise: Two Confusion Matrices

Both matrices have the same percent agreement (82%).
Which should have higher κ, and why?

Matrix A

	B: Pos	B: Neg
A: Pos	41	9
A: Neg	9	41

Marginals: 50/50 each

Matrix B

	B: Pos	B: Neg
A: Pos	78	4
A: Neg	14	4

Marginals: very skewed

This sets up today's theme: prevalence, bias, and the "κ paradox"—all amplified in LLM settings.

LLMs Can Annotate Competitively

Recent findings (2023–2026):

- ChatGPT outperforms crowdworkers on multiple text-annotation tasks (Gilardi et al., PNAS 2023)
- GPT-4 matches or exceeds expert baselines on political-affiliation labeling (Törnberg, 2023)
- Detailed prompts with exemplars improve LLM reliability to expert levels (Oelen et al., Nature Machine Intelligence 2026)

But: reliability depends on prompt design, task domain, fine-tuning, and model version.

Caveat

High accuracy on one dataset $\neq$ universal reliability.

What Changes When the Annotator Is an LLM?

Three new sources of variation that humans don't have:

1. **Prompt wording** — different instructions → different labels
   "Treat a *prompt variant* as a different annotator."
2. **Decoding randomness** — temperature > 0 makes outputs stochastic
   Same prompt, same item → different labels across runs
3. **Model snapshot / platform changes** — the model itself changes over time
   Backend updates can silently alter behavior

Implication

You are no longer measuring reliability among **humans** alone, but among a mixed set of **humans, prompts, model snapshots, decoding settings**, and sometimes **repeated samples from stochastic generators**.

Non-Determinism and Moving Targets

LLM outputs can vary across calls even with identical prompts.

- Temperature > 0: outputs are explicitly stochastic
- Temperature $= 0$: still not guaranteed deterministic (platform-level changes)
- Official guidance recommends logging:
  - Random **seed**
  - **System fingerprint** to detect backend changes

Consequence for agreement:

- If you sample the same LLM K times on the same N items, each run is an "annotator replicate"
- You can compute pairwise κ across runs $\rightarrow$ **test–retest reliability**

What Is a System Fingerprint?

A system fingerprint is:

- Generated by the **model provider**, returned in response metadata
- A hash representing the **deployed model + infrastructure state**
- **Not** something you can reverse-engineer or generate locally

You just log it.

Analogy

In traditional annotation, we log **annotator ID**.
In LLM annotation, we log **annotator ID + decoding parameters + backend fingerprint**.

Otherwise you are not logging your measurement instrument.

Getting the Fingerprint in Practice

Case A: Using an API (recommended for research)

```
response = client.responses.create(...)
print(response.system_fingerprint)
# e.g. "fp_abc123xyz"
```

Log with every run: model name, temperature, seed, system fingerprint, prompt version, timestamp.

Case B: Web chat interface

- Fingerprints are **not accessible** → cannot guarantee reproducibility
- Serious annotation studies should **not** rely on manual chat UI calls

Case C: Open-source local models (LLaMA, Mistral, etc.)

- No provider fingerprint — instead log: model checkpoint hash, repo commit, tokenizer version, inference library version, hardware details
- Your **environment snapshot** *is* the fingerprint

Why Fingerprints Matter

Scenario:

1. You annotate 10,000 items in January
2. In March, the provider silently updates model weights
3. Your κ drops by 0.07

Without fingerprint logging, you cannot distinguish:

- Sampling noise?
- Model weight change?
- Safety-filtering update?

Important subtlety

Even with temperature $= 0$, fixed seed, and same fingerprint, you are **still not guaranteed determinism**. Parallelism, floating-point non-determinism, and distributed inference can all introduce variation.

Fingerprints detect **backend version changes**, not stochastic variation.

Prompt-Dependence = “Annotator Drift”

In LLM-judge studies:

- Prompt variants measurably change agreement with humans
- More instruction is not guaranteed to increase alignment
- There can be systematic tendencies (e.g., inflated high scores)

Example: An LLM grading essays with prompt A gives mean score 4.2/5. The same LLM with prompt B gives mean score 3.6/5.

Teaching point

Treat the **LLM configuration** (model + prompt + temperature + seed) as part of the measurement instrument, not just the model itself.

Prevalence / Bias Sensitivity with LLMs

Recall the "κ paradox": κ can drop sharply under extreme class imbalance even when percent agreement is high.

This is amplified with LLMs because LLM labeling often creates or amplifies imbalance.

Positive prevalence	Percent agreement (P_o)	Cohen's κ
0.01	0.82	0.066
0.05	0.82	0.252
0.10	0.82	0.390
0.20	0.82	0.532
0.50	0.82	0.640
0.80	0.82	0.532
0.90	0.82	0.390

Same $P_o = 0.82$, wildly different κ!

Summary: What Breaks with LLM Annotators

Classic assumption	What changes with LLMs
Fixed annotator	Model updates, prompt variants, temperature
Independence	LLMs trained on overlapping data; correlated errors
Stable marginals	Prompt changes shift label distributions
Hard labels	LLMs can output probabilities / confidence scores

Bottom line: Classic coefficients (κ, α) remain foundational, but students must learn to:

(a) Treat the LLM configuration as part of the measurement instrument

(b) Diagnose when agreement numbers are inflated/deflated

(c) Compute and interpret agreement when outputs are *probabilistic*

LLM as Annotator: Three Approaches

Goal: produce parseable outputs and (optionally) probabilities.

Why structured outputs?

- Schema-constrained JSON responses reduce parsing noise
- Makes annotation pipelines reproducible
- Official API guidance supports this

Three ways to get labels from an LLM:

1. **Hard labels** — single categorical output, compute κ directly
2. **JSON + confidence** — structured output with self-reported certainty
3. **Logprobs** — actual class probabilities from the model's output distribution

We ran all three on a small stance-detection task. Let's walk through the results.

The Example Set: Stance Detection

Labels: Support / Oppose / Neutral (5 items, 2 human annotators, 2 LLM configs)

#	Text	Ambiguity
1	"Great—another 2% fee, exactly what we needed."	Sarcasm
2	"I'm not against it; I'm against *how* they're doing it."	Mixed stance
3	"This is a complicated issue with tradeoffs."	Hidden stance
4	"Finally, someone is taking action."	Vague target
5	"Stop pretending this helps ordinary people."	Target unclear

Key point: every item has a reason why annotators might disagree—this is by design.

Introduces: *label-set adequacy* and *guideline precision* as drivers of disagreement.

Hard Labels: The Prompts

Prompt A (minimal):

```
System: You are a careful annotator. Follow the labeling
        guide exactly. Output only the label.
User:   Label set: {Support, Oppose, Neutral}.
        Guide: "Support" = in favor of the policy;
        "Oppose" = against the policy;
        "Neutral" = no clear stance.
        Text: "Great -- another 2% fee, exactly what
        we needed."
```

Prompt B (one sentence added to the user message):

```
        ...
        "Neutral" = no clear stance.
        Watch for sarcasm and irony. If the literal
        wording contradicts the likely intent, label
        based on intent.
        Text: "Great -- another 2% fee, exactly what
        we needed."
```

The only difference: one instruction about sarcasm. Let's see what that does.

Hard Labels: Results

#	Human 1	Human 2	LLM (Prompt A)	LLM (Prompt B)
1	Oppose	Oppose	Support	Oppose
2	Oppose	Neutral	Oppose	Neutral
3	Neutral	Neutral	Neutral	Neutral
4	Support	Support	Support	Support
5	Oppose	Oppose	Oppose	Oppose

Pairwise Cohen's κ:

	H1	H2	LLM-A	LLM-B
H1	1.00	0.74	0.52	0.74
H2		1.00	0.22	1.00
LLM-A			1.00	0.22
LLM-B				1.00

Notice: Prompt A misreads the sarcasm in Item 1. One sentence of instruction (Prompt B) fixes it. **Prompt = instrument.**

JSON + Confidence: The Prompt

Full prompt:

```
System: You are an expert annotator. Return JSON with
        fields: label (one of [Support, Oppose, Neutral])
        and confidence (0-1). No extra keys.
User:   Guide: "Support" = in favor of the policy;
        "Oppose" = against the policy;
        "Neutral" = no clear stance.
        Watch for sarcasm and irony. If the literal
        wording contradicts the likely intent, label
        based on intent.
        Text: "Great -- another 2% fee, exactly what
        we needed."
```

Example output:

```
{"label": "Oppose", "confidence": 0.72}
```

The LLM returns a single hard label *plus* a self-reported certainty score.

JSON + Confidence: Results

#	LLM Label	Confidence	Note
1	Support	0.85	Wrong and overconfident
2	Oppose	0.72	Matches H1, reasonable
3	Neutral	0.91	Matches both, *too* confident?
4	Support	0.95	Correct and confident
5	Oppose	0.88	Correct and confident

Key lesson

Item 1: the LLM is **85% confident** in the **wrong** label.

Self-reported confidence is often **miscalibrated** in LLMs—use it as a teaching artifact for calibration, not as ground truth.

Logprobs: The Prompt

Prompt (constrains output to a single token):

```
System: You are a classifier. Output exactly one token:
        A, B, or C.
User:   A = Support, B = Oppose, C = Neutral.
        Guide: "Support" = in favor of the policy;
        "Oppose" = against the policy;
        "Neutral" = no clear stance.
        Watch for sarcasm and irony. If the literal
        wording contradicts the likely intent, label
        based on intent.
        Text: "Great -- another 2% fee, ..."
```

Why A/B/C instead of "Support" / "Oppose" / "Neutral"?

- `top_logprobs` returns probabilities for the **first token** only
- "Support" may tokenize as `Sup+port` — logprobs would be incomplete
- A, B, C are **guaranteed single tokens** → full distribution in one call

Logprobs: API Setup

Logprobs come from the API parameter, not the prompt:

```
response = client.chat.completions.create(
    ..., max_tokens=1, logprobs=True, top_logprobs=3)

# extract logprobs for A, B, C from response
logprob_A = ...  # returned in response.choices[0]
logprob_B = ...  #   .logprobs.content[0].top_logprobs
logprob_C = ...
```

Then convert to a probability distribution (softmax):

$$p(\text{Support}) = \frac{e^{\text{logprob}(A)}}{\sum_{x \in \{A,B,C\}} e^{\text{logprob}(x)}}$$

Key idea

The prompt constrains the output to one token.
The API parameter extracts what the model "thought" about each option.

Logprobs: Results

#	p(Support)	p(Oppose)	p(Neutral)	Argmax
1	**0.47**	0.41	0.12	Support
2	0.08	**0.65**	0.27	Oppose
3	0.05	0.11	**0.84**	Neutral
4	**0.89**	0.03	0.08	Support
5	0.02	**0.91**	0.07	Oppose

Compare Items 1 and 4:

- Item 4: $p(\text{Support}) = 0.89$ — the model is decisive
- Item 1: $p(\text{Support}) = 0.47$ vs. $p(\text{Oppose}) = 0.41$ — **nearly a coin flip**

The hard label says "Support" for both. The distribution reveals that Item 1 is genuinely ambiguous.

This is exactly what soft metrics capture.

Agreement $\neq$ Validity

Key insight

LLM–LLM agreement can be **very high** while still being systematically **wrong** or **biased toward one score/label**.

High LLM–LLM κ does NOT mean:

- The labels are correct
- The LLM agrees with humans
- The annotation scheme is valid

Agreement $\neq$ validity.

Two LLMs trained on similar data can produce *correlated* errors, inflating agreement without any guarantee of correctness.

LLM Stability: Repeat Runs = "Replicate Annotators"

Call the the same model K times on the same N items:

```
labels_runs = [
    ["A","B","B","C","A"],  # run 1
    ["A","B","B","C","A"],  # run 2 (temp 0 might match)
    ["A","B","C","C","A"],  # run 3 (higher temp drifts)
]

from itertools import combinations
from sklearn.metrics import cohen_kappa_score

for i, j in combinations(range(len(labels_runs)), 2):
    print(i, j, cohen_kappa_score(
        labels_runs[i], labels_runs[j]))
```

Tie-in: Some evaluation setups report high consistency at temperature 0, but that does not guarantee *human alignment* or absence of systematic score inflation.

Why Hard Labels Are Not Enough

3 classes (A/B/C): all items match by argmax → hard agreement looks perfect.

Item	Human dist.	LLM dist.	Hard match?	Soft match $\mathbf{p}\cdot\mathbf{q}$	JSD
1	(0.90, 0.10, 0.00)	(0.60, 0.40, 0.00)	✓	0.580	0.302
2	(0.50, 0.50, 0.00)	(0.90, 0.10, 0.00)	✓	0.500	0.383
3	(0.34, 0.33, 0.33)	(0.80, 0.10, 0.10)	✓	0.338	0.403
4	(0.10, 0.90, 0.00)	(0.05, 0.95, 0.00)	✓	0.860	0.081

Hard agreement = 100%, but distributional agreement varies wildly!

This is why two recent projects matter:

- **ChaosNLI** (Nie et al., 2020): re-collected NLI labels from **100 annotators** per item. Found that many "gold" labels mask genuine disagreement → evaluate against *distributions*, not single labels.
- **UNLI** (Chen et al., 2020): annotators assign a **probability** (0–1) instead of a category. Annotation itself becomes probabilistic → the "right" target is a distribution, not a single label.

Key Takeaways

1. **LLM annotation = measurement with a moving instrument**
   Prompt, temperature, model snapshot all matter
2. **Classic κ/α still work**, but watch for prevalence/bias artifacts with LLMs
3. **High LLM–LLM agreement $\neq$ validity**
   Correlated errors inflate agreement without guaranteeing correctness
4. **Reproducibility** requires logging model, prompt, temperature, seed

Questions?

Office Hours: Wednesdays 1-3pm, Volen 109

jinzhao@brandeis.edu