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

## Lecture 12: Modeling Annotator Reliability and Label Noise

Jin Zhao

Brandeis University

March 4, 2026

# Today's Agenda

1. Continued from last time: soft agreement & JSD
2. Warm-up: when majority vote fails + expert adjudication
3. The Dawid–Skene model: latent true labels + confusion matrices
4. Dawid–Skene EM: results on toy data
5. Majority vote vs. DS under controlled noise
6. Soft labels for training + mixed human/LLM annotators
7. LLM-assisted adjudication
8. Creating gold standard datasets + documentation
9. Key takeaways

# Soft Agreement: The Core Idea

**With hard labels:** two annotators either *match* or *don't*.

**With distributions:** we ask *how much* do they overlap?

| | Support | Oppose | Neutral |
|---|---|---|---|
| Human crowd | 0.20 | 0.70 | 0.10 |
| LLM output | 0.10 | 0.80 | 0.10 |

Both say "Oppose" — hard labels match. But *how similar* are the full distributions?

**Soft agreement for one item** = multiply matching entries and add up:

$$\underbrace{(0.20 \times 0.10)}_{\text{Support}} + \underbrace{(0.70 \times 0.80)}_{\text{Oppose}} + \underbrace{(0.10 \times 0.10)}_{\text{Neutral}} = 0.02 + 0.56 + 0.01 = \mathbf{0.59}$$

**Why this works:** when both distributions put mass in the same places, the products are large and the sum is high. When they disagree, the products shrink.

# Soft Agreement: A Worked Example

**Two items, two annotators (Human crowd vs. LLM):**

| # | Annotator | Sup | Opp | Neu | Per-item soft agreement |
|---|---|---|---|---|---|
| 1 | Human | 0.20 | 0.70 | 0.10 | $0.20\times0.10 + 0.70\times0.80 + 0.10\times0.10 = \mathbf{0.59}$ |
| 1 | LLM | 0.10 | 0.80 | 0.10 | |
| 2 | Human | 0.90 | 0.05 | 0.05 | $0.90\times0.85 + 0.05\times0.05 + 0.05\times0.10 = \mathbf{0.77}$ |
| 2 | LLM | 0.85 | 0.05 | 0.10 | |

**Step 1: Average across items** to get overall soft observed agreement:

$$P_o^{\text{soft}} = \frac{0.59 + 0.77}{2} = \mathbf{0.68}$$

That's it — $P_o^{\text{soft}}$ is just the **average of the per-item overlaps**.

# Soft Agreement: Correcting for Chance

**Recall:** $H_1$: .20 .70 .10 $H_2$: .90 .05 .05 $P_o^{\text{soft}} = 0.68$
$L_1$: .10 .80 .10 $L_2$: .85 .05 .10

**Step 2: What overlap would we expect by chance?** Average each annotator's distributions across items:

| | Sup | Opp | Neu |
|---|---|---|---|
| Human (avg) | $\frac{0.20+0.90}{2} = 0.55$ | $\frac{0.70+0.05}{2} = 0.375$ | $\frac{0.10+0.05}{2} = 0.075$ |
| LLM (avg) | $\frac{0.10+0.85}{2} = 0.475$ | $\frac{0.80+0.05}{2} = 0.425$ | $\frac{0.10+0.10}{2} = 0.10$ |

Chance overlap = multiply and add (same operation):

$$P_e^{\text{soft}} = (0.55 \times 0.475) + (0.375 \times 0.425) + (0.075 \times 0.10) = \mathbf{0.428}$$

**Step 3: Soft** $\kappa$ (same formula as regular $\kappa$):

$$\kappa^{\text{soft}} = \frac{0.68 - 0.428}{1 - 0.428} = \frac{0.252}{0.572} = \mathbf{0.44}$$

**Interpretation:** moderate agreement beyond chance — the distributions overlap, but not perfectly.

# Jensen–Shannon Distance (JSD): The Intuition

**Soft $\kappa$ asks:** "how much do distributions overlap, beyond chance?"
**JSD (Jensen–Shannon Distance) asks a different question:** "how *far apart* are two distributions?"

**The idea in 3 steps:**

1. **Blend** the two distributions into a midpoint: $\mathbf{m} = \frac{\mathbf{p}+\mathbf{q}}{2}$
2. **Measure** how "surprised" each original distribution is by the blend
   (this uses KL divergence — we'll compute it with plain arithmetic)
3. **Average** the two surprise values $\rightarrow$ that's JSD

| | Support | Oppose | Neutral |
|---|---|---|---|
| Human ($\mathbf{p}$) | 0.60 | 0.30 | 0.10 |
| LLM ($\mathbf{q}$) | 0.85 | 0.10 | 0.05 |
| Midpoint ($\mathbf{m}$) | $\frac{0.60+0.85}{2} = 0.725$ | $\frac{0.30+0.10}{2} = 0.20$ | $\frac{0.10+0.05}{2} = 0.075$ |

Both say "Support" — but *how* different are the full pictures?

# JSD: Worked Example (Step by Step)

**KL divergence** = for each label, compute: value $\times \ln\left(\frac{\text{value}}{\text{midpoint}}\right)$, then add up.

**Human $\rightarrow$ midpoint:**

| | Support | Oppose | Neutral | Sum |
|---|---|---|---|---|
| $p_k \times \ln(p_k/m_k)$ | $0.60 \times \ln\frac{0.60}{0.725}$ | $0.30 \times \ln\frac{0.30}{0.20}$ | $0.10 \times \ln\frac{0.10}{0.075}$ | |
| | $= -0.115$ | $= 0.122$ | $= 0.029$ | **0.036** |

**LLM $\rightarrow$ midpoint:**

| | Support | Oppose | Neutral | Sum |
|---|---|---|---|---|
| $q_k \times \ln(q_k/m_k)$ | $0.85 \times \ln\frac{0.85}{0.725}$ | $0.10 \times \ln\frac{0.10}{0.20}$ | $0.05 \times \ln\frac{0.05}{0.075}$ | |
| | $= 0.133$ | $= -0.069$ | $= -0.020$ | **0.044** |

**JSD** = average: $\frac{0.036+0.044}{2} = \mathbf{0.040}$

**JS Distance** $= \sqrt{0.040} = \mathbf{0.20}$ (on a 0–1 scale)

# JSD: What the Number Means

**JS Distance is on a 0–1 scale:**

| JS Distance | Interpretation |
|---|---|
| 0.00 | Identical distributions |
| $\approx 0.10$ | Very similar |
| $\approx 0.20$ | Moderate difference $\leftarrow$ our example |
| $\approx 0.40$ | Substantial difference |
| 1.00 | Maximally different (no overlap) |

**Our example:** both say "Support," but the LLM is much more confident (0.85 vs. 0.60). JS Distance $= 0.20$ captures that difference; hard labels miss it entirely.

This is the metric used in the **ChaosNLI** paper for distributional evaluation.

# Beyond $\kappa$: What Else to Report

**$\kappa$ alone doesn't tell the whole story. Always report these alongside it:**

1. **Class distribution** — show how often each label was used
   If one label dominates (e.g., 95% Neutral), $\kappa$ will be deflated even if agreement is high
2. **Per-class agreement** — break $\kappa$ down by label
   "Overall $\kappa = 0.6$" can hide that Support has $\kappa = 0.9$ while Neutral has $\kappa = 0.2$
3. **Sensitivity across prompts / temperatures**
   Run the same task with two prompts or two temperature settings and compare $\kappa$ — if it changes a lot, your results depend on configuration, not just the data
4. **A distributional metric** (soft $\kappa$ or JSD)
   Especially when you have logprobs or crowd distributions available

# High LLM $\kappa$ with Experts $\neq$ Problem Solved

**Scenario:** An LLM achieves $\kappa = 0.85$ with expert annotators—higher than crowdworkers ($\kappa = 0.65$).

**What this proves:**

- The LLM's labels are *consistent* with expert labels on this dataset

**What this does NOT prove:**

- That the LLM "understands" the task
- That it will generalize to new domains or edge cases
- That the labels are *valid* (experts can be wrong too)

**Annotation leakage**

If the LLM was trained on data that includes the annotation guidelines or the labeled dataset itself, high agreement may reflect **memorization**, not annotation skill.

# Should We Maximize Agreement on Subjective Tasks?

**Not always.** Some tasks have *genuine* disagreement.

**Example:** "Is this tweet offensive?"

- 60% of annotators say yes, 40% say no
- This split is not noise—it reflects real differences in perspective

**Forcing a single gold label erases meaningful information.**

**Two approaches:**

1. **Majority vote** — simple, but throws away the 40% minority view
2. **Preserve the distribution** — keep the (0.60, 0.40) split as the target
   This is what ChaosNLI advocates: evaluate models against the *distribution*, not a single label

**Key insight:** low agreement on subjective tasks is *expected*, not a red flag.
Low agreement on objective tasks (e.g., part-of-speech tagging) *is* a red flag.

# Consistent but Wrong: The LLM Self-Agreement Trap

**Scenario:** Your LLM judge scores essays. Across 5 runs, it gives nearly identical scores (self-$\kappa = 0.98$). But its scores differ from human experts ($\kappa = 0.45$).

**What's happening?**

- The LLM is **consistent** (high self-agreement)
- But it's consistently **biased** (e.g., inflating scores, ignoring certain criteria)
- Consistency $\neq$ correctness

**What to do:**

1. Always compute **human–LLM** agreement, not just LLM–LLM
2. Check for **systematic bias**: does the LLM consistently favor certain labels?
3. Compare the LLM's **label distribution** to the human distribution
   If humans are 40/40/20 but the LLM is 70/20/10, that's a bias problem

This ties back to our earlier point: **agreement** $\neq$ **validity**.

# Reproducibility Checklist for LLM Annotation

**What to require in student work (and in published research):**

1. **Log:** model identifier/snapshot, prompt, temperature, seed + system fingerprint
2. **Use structured outputs:** JSON schema to avoid parsing ambiguity
3. **Cache raw model responses:** so the same run can be re-scored without re-querying
4. **Report calibration diagnostics:** when using confidence scores, warn about miscalibration
5. **Report multiple metrics:** $\kappa$ + at least one distributional/soft metric

**Data governance:** sending text to third-party LLM APIs is a data-sharing decision. Avoid sensitive personal data in classroom demos.

# The Core Mental Model

**Last lecture:** we measured agreement. Today we go further:

**Central idea**

Observed labels are not ground truth.

They are **noisy measurements** produced by annotators with different error profiles.

**Agreement is a symptom. Today we model the cause.**

Instead of just computing agreement scores, we will:

1. Estimate each annotator's **error profile** (confusion matrix)
2. Infer the **latent true label** for each item
3. Use this to produce **better labels** than majority vote

# Warm-up: Should We Believe the Majority?

**Scenario:** 3 annotators, 1 item, binary task (Positive / Negative).

| Annotator | Label | Background |
|---|---|---|
| A | Positive | Expert (5 years experience) |
| B | Negative | Novice (first week) |
| C | Negative | Novice (first week) |

**Majority vote says: Negative** (2 vs. 1).

**But should we trust two novices over one expert?**

**The problem with majority vote**

Every annotator gets **equal weight**, regardless of reliability.
This works when annotators are similarly competent.
It fails when they differ in quality — which is common in practice.

# When Majority Vote Works vs. When It Fails

**Majority vote is a strong baseline when:**

- Annotators are **similarly competent** and errors are roughly symmetric
- You have **many annotators** per item (errors cancel out)

**Majority vote breaks down when:**

- Annotators **differ in reliability** (experts mixed with novices)
- Some annotators are **systematically biased** (e.g., always label Positive)
- Labeling is **sparse** (few labels per item — can't form a majority)
- A **spammer** contributes random labels that dilute expert signal

**Key insight**

We need a method that **learns who is reliable** and weights annotators accordingly.
That method is **Dawid–Skene**.

# Expert Adjudication

**For higher quality gold standard**

**Process:**

1. Identify items with disagreement
2. Expert reviews each case
3. Expert makes final decision
4. Optionally: update guidelines based on patterns

**When to use:**

- High-stakes tasks
- Creating evaluation benchmarks
- Complex annotation requiring domain expertise

**Cost:** More expensive and slower, but produces higher-quality labels

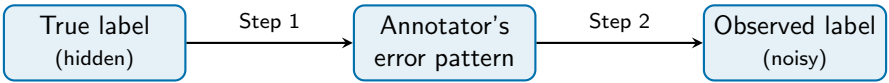
# The Dawid–Skene Model: The Idea

**Dawid & Skene (1979):** one of the most cited papers in annotation methodology.

**The setup (in plain English):**

- We have $N$ items to label (e.g., 80 sentences)
- Each item has a **true label** we don't know — that's what we want to find!
- We have $M$ annotators, each with their own **error pattern**
- We only see the annotators' (noisy) labels

**The "story" of how labels get generated:**




**Step 1:** Nature picks a true label (e.g., Pos, Neg, or Neu).
**Step 2:** Each annotator "looks through" their personal error pattern and produces a (possibly wrong) label.

**Goal:** reverse-engineer the true labels from the noisy observations.

# DS: What Happens to One Item

**True label = Pos.** Three annotators each label this item:

**Error pattern** = across all items whose true label is Pos, how often does this annotator say each class?

| Annotator | Error pattern: Says Pos | Says Neg | Says Neu | **Result** |
|---|---|---|---|---|
| Expert | **92%** | 4% | 4% | Says **Pos** ✓ |
| Biased | 30% | **50%** | 20% | Says **Neg** × |
| Spammer | 33% | 33% | 33% | Says **Neu** × |

**We observe:** Pos, Neg, Neu — three different answers! But the true label is Pos.

## This is what DS models

Each annotator has their own error pattern (how likely they are to say each label).
DS **learns these patterns from data** and uses them to figure out the true label.

# What Is a Confusion Matrix? (Annotator Version)

**For a 3-class task** (Pos, Neg, Neu), each annotator has a table like this:

| | Says Pos | Says Neg | Says Neu | |
|---|---|---|---|---|
| True = Pos | **??%** | ??% | ??% | row sums to 100% |
| True = Neg | ??% | **??%** | ??% | row sums to 100% |
| True = Neu | ??% | ??% | **??%** | row sums to 100% |

**Diagonal** (bold) = **correct answers.** Off-diagonal = errors.

**A perfect annotator:** diagonal is all 100%, off-diagonal all 0%.
**A spammer:** every row looks the same (labels are random).
**A biased annotator:** one column has high values across all rows (always says "Neg").

**Key idea:** the confusion matrix is an annotator's **signature** — it captures their accuracy, biases, and typical mistakes.

# Example: Four Annotator Profiles

**3-class sentiment task. Four annotators with very different profiles:**

**Expert** (high diagonal):

| | Pos | Neg | Neu |
|---|---|---|---|
| Pos | **0.92** | 0.04 | 0.04 |
| Neg | 0.04 | **0.92** | 0.04 |
| Neu | 0.04 | 0.04 | **0.92** |

**Decent** (moderate diagonal):

| | Pos | Neg | Neu |
|---|---|---|---|
| Pos | **0.78** | 0.11 | 0.11 |
| Neg | 0.11 | **0.78** | 0.11 |
| Neu | 0.11 | 0.11 | **0.78** |

**Biased** (overuses Neg):

| | Pos | Neg | Neu |
|---|---|---|---|
| Pos | 0.30 | **0.50** | 0.20 |
| Neg | 0.10 | **0.70** | 0.20 |
| Neu | 0.10 | **0.60** | 0.30 |

**Spammer** (uniform rows):

| | Pos | Neg | Neu |
|---|---|---|---|
| Pos | 0.33 | 0.33 | 0.33 |
| Neg | 0.33 | 0.33 | 0.33 |
| Neu | 0.33 | 0.33 | 0.33 |

**DS learns these matrices from data — without knowing who is expert or spammer.**

# The DS Assumption: Errors Are Independent

**Key assumption:** given the true label, each annotator makes mistakes **on their own**.

**Concrete example:** suppose the true label for Item 5 is Pos.

- Expert: 92% chance of saying Pos (regardless of what others do)
- Biased: 30% chance of saying Pos (regardless of what others do)
- Spammer: 33% chance of saying Pos (regardless of what others do)

Each annotator's label depends **only** on the true label and their own error pattern — **not** on what other annotators said.

**Realistic when:**

- Annotators work independently
- Different kinds of mistakes

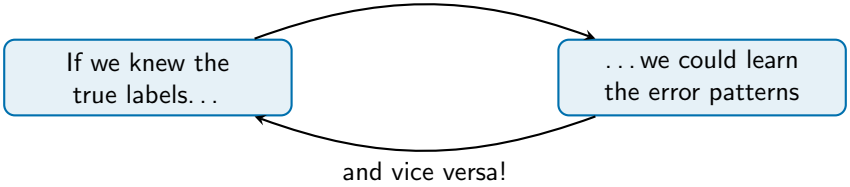
**Violated when:**

- Annotators discuss items
- Same LLM prompt template
- Crowdworkers copy each other

Treat this as a useful baseline assumption, not a law.

# How DS Learns: The Chicken-and-Egg Problem

**Problem:** we don't know the true labels *or* the error patterns. Chicken and egg!




**The EM trick:** alternate between the two steps!

1. **Start** with a rough guess: use majority vote as initial "true labels"
2. **Learn patterns:** given the guessed labels, count each annotator's hits and misses
3. **Re-guess labels:** given the learned patterns, re-evaluate which label is most likely
4. **Repeat** steps 2–3 until nothing changes

This is called **Expectation-Maximization (EM)** — a general technique for learning with missing data.

# EM in NLP: A Brief History

**Classic NLP (2000–2015)**

- POS tagging → HMMs (Baum–Welch)
- Word alignment → IBM Models / GIZA++
- Unsupervised parsing → PCFGs (inside–outside)
- Topic models → LDA (variational EM)
- Annotation → Dawid–Skene ← **we are here**

**Why less common now**

- Deep learning replaced generative models (HMM → BiLSTM, IBM alignment → attention)
- EM needs exact expectations — intractable for neural nets
- Large labeled datasets reduce need for unsupervised EM

**Still used today**

- Annotation aggregation (DS, MACE)
- Topic models (LDA)

# EM Walkthrough: The Setup

**Item 7:** Expert says Pos, Biased says Neg, Spammer says Neg.

**Majority vote:** Neg (2 vs. 1). But is that right?

**DS idea:** try each possible true label, see how well it explains what we observed.

For each candidate true label, multiply three things:

1. **Class prior** — how common is this class? (from MV initialization: 45% / 35% / 20%)
2. **P(Expert says Pos | true label)** — Expert's *likelihood* (row = true label, col = P)
3. **P(Biased says Neg | true label)** — Biased's *likelihood* (row = true label, col = N)
4. **P(Spammer says Neg | true label)** — Spammer's *likelihood* (row = true label, col = N)

**Confusion matrices (rows = true label, cols = says):**

| **Expert** | P | N | U |
|---|---|---|---|
| P | .92 | .04 | .04 |
| N | .04 | .92 | .04 |
| U | .04 | .04 | .92 |

| **Biased** | P | N | U |
|---|---|---|---|
| P | .30 | .50 | .20 |
| N | .10 | .70 | .20 |
| U | .10 | .60 | .30 |

| **Spammer** | P | N | U |
|---|---|---|---|
| P | .33 | .33 | .33 |
| N | .33 | .33 | .33 |
| U | .33 | .33 | .33 |

# EM Walkthrough: What If True = Pos?

**Assume the true label is Pos.** Look up each annotator's number:

| Annotator | Said | Where to look (slide 20) | Value |
|---|---|---|---|
| Expert | Pos | True=Pos row, Says Pos col | **0.92** (very likely!) |
| Biased | Neg | True=Pos row, Says Neg col | **0.50** (often says Neg anyway) |
| Spammer | Neg | True=Pos row, Says Neg col | **0.33** (random) |

**Multiply with class prior:**

$$\underbrace{0.45}_{\text{prior}} \times \underbrace{0.92}_{\text{Expert}} \times \underbrace{0.50}_{\text{Biased}} \times \underbrace{0.33}_{\text{Spammer}} = \mathbf{0.0683}$$

This score measures: *how well does "true = Pos" explain what we observed?*

*True=Pos row:* Expert **.92**/.04/.04 Biased .30/**.50**/.20 Spammer .33/**.33**/.33

# EM Walkthrough: What If True = Neg or Neu?

**Same lookup for the other two candidates:**

**True = Neg:** Expert says Pos | true=Neg → **0.04** (very unlikely!)

$$0.35 \times 0.04 \times 0.70 \times 0.33 = \mathbf{0.0032}$$

**True = Neu:** Expert says Pos | true=Neu → **0.04** (also very unlikely!)

$$0.20 \times 0.04 \times 0.60 \times 0.33 = \mathbf{0.0016}$$

**Key insight:** the Expert almost never says Pos when the truth isn't Pos — only 4% of the time.

So the Expert saying Pos is very strong evidence the truth **is** Pos, enough to override the 2-to-1 majority vote.

*True=Neg row:* Expert **.04**/.92/.04 Biased .10/**.70**/.20 Spammer .33/**.33**/.33

*True=Neu row:* Expert **.04**/.04/.92 Biased .10/**.60**/.30 Spammer .33/**.33**/.33

# EM Walkthrough: The Final Answer

**Normalize the three scores to get probabilities:**

| True label | Raw score | Normalized |
|---|---|---|
| Pos | 0.0683 | **93%** |
| Neg | 0.0032 | 4% |
| Neu | 0.0016 | 2% |

**DS says Pos!** Majority vote said Neg — **DS overruled it.**

**Why?** The Expert's "Pos" is 23× more likely if true=Pos vs. true=Neg:

$$\frac{P(\text{Expert says Pos} \mid \text{Pos})}{P(\text{Expert says Pos} \mid \text{Neg})} = \frac{0.92}{0.04} = 23$$

After more EM iterations (re-estimating confusion matrices → re-computing scores → repeat), DS converges to stable estimates.

# The E-Step: Posteriors for Every Item

**What slides 25–27 just did is called the E-step** (E = Expectation).

**Recipe:** for each item, multiply prior × likelihoods from every annotator, then normalize.

DS does this for **all 80 items**, not just Item 7:

| Item | Expert | Biased | Spammer | $P$(Pos) | $P$(Neg) | $P$(Neu) |
|---|---|---|---|---|---|---|
| 3 | Neg | Neg | Pos | 0.10 | 0.85 | 0.05 |
| 5 | Pos | Neg | Pos | 0.80 | 0.15 | 0.05 |
| 7 | Pos | Neg | Neg | **0.93** | 0.04 | 0.02 |
| 12 | Neg | Neg | Neg | 0.05 | 0.90 | 0.05 |
| 18 | Pos | Pos | Neu | 0.90 | 0.03 | 0.07 |

**Key point**

The E-step produces **soft labels** (probability distributions), not hard labels.

# The M-Step: Re-estimate Confusion Matrices

Given posteriors from the E-step, update confusion matrices using **soft counts**.

**Worked example:** compute one cell

$P(\text{Expert says Pos} \mid \text{true} = \text{Pos})$

(5 items shown; in practice, sum over all 80.)

| Item | Expert said | $P(\text{t} = \text{P})$ | Said P? |
|---|---|---|---|
| 5 | Pos | 0.80 | ✓ |
| 7 | Pos | 0.93 | ✓ |
| 12 | Neg | 0.05 | |
| 18 | Pos | 0.90 | ✓ |
| 3 | Neg | 0.10 | |

**Expert's matrix**

| | Says P | Says N | Says U |
|---|---|---|---|
| True P | **0.95** | ? | ? |
| True N | ? | ? | ? |
| True U | ? | ? | ? |

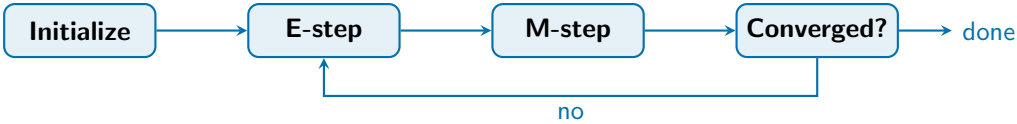
Repeat for every cell, every annotator.

$$\underbrace{0.80 + 0.93 + 0.90}_{\text{said Pos, weighted}} = 2.63 \quad / \quad \underbrace{0.80 + 0.93 + 0.05 + 0.90 + 0.10}_{\text{all items, weighted}} = 2.78 \quad = \quad \mathbf{0.95}$$

# The M-Step: Re-estimate Priors + The Full Loop

**Re-estimate priors:** average the posteriors across **all** items. (5 shown here; in practice, sum over all 80.)

$$P(\text{Pos}) = \frac{0.10 + 0.80 + 0.93 + 0.05 + 0.90}{5} = \frac{2.78}{5} = 0.56 \quad P(\text{Neg}) = 0.38 \quad P(\text{Neu}) = 0.06$$




Typically **5–20 iterations**; first 2–3 do most of the work. Stop when posteriors change by $< \epsilon$ (e.g. $10^{-4}$) between iterations.

## Summary

**E-step** = "given matrices, what are the labels?" **M-step** = "given labels, what are the matrices?"

# Experiment Setup: Synthetic Annotation Data

**We simulated a realistic annotation scenario:**

- **80 items**, 3 classes (Pos, Neg, Neu)
- True class distribution: 45% Pos, 35% Neg, 20% Neu
- **4 annotators** with different reliability profiles
- Each item labeled by **3 randomly chosen** annotators (sparse)

**Ground-truth confusion matrices** (used to generate the data — DS does **not** see these):

**Expert**

| | P | N | U |
|---|---|---|---|
| P | **.92** | .04 | .04 |
| N | .04 | **.92** | .04 |
| U | .04 | .04 | **.92** |

**Decent**

| | P | N | U |
|---|---|---|---|
| P | **.78** | .11 | .11 |
| N | .11 | **.78** | .11 |
| U | .11 | .11 | **.78** |

**Biased**

| | P | N | U |
|---|---|---|---|
| P | .30 | **.50** | .20 |
| N | .10 | **.70** | .20 |
| U | .10 | **.60** | .30 |

**Spammer**

| | P | N | U |
|---|---|---|---|
| P | .33 | .33 | .33 |
| N | .33 | .33 | .33 |
| U | .33 | .33 | .33 |

**The key question:** can DS recover these profiles and the true labels better than majority vote?

# DS Results: Learned Confusion Matrices

**After running EM, DS recovers the annotator profiles:**

**Expert** (learned):

| | Pos | Neg | Neu |
|---|---|---|---|
| Pos | **0.91** | 0.05 | 0.04 |
| Neg | 0.03 | **0.93** | 0.04 |
| Neu | 0.05 | 0.03 | **0.92** |

**Decent** (learned):

| | Pos | Neg | Neu |
|---|---|---|---|
| Pos | **0.76** | 0.13 | 0.11 |
| Neg | 0.10 | **0.79** | 0.11 |
| Neu | 0.12 | 0.10 | **0.78** |

**Biased** (learned):

| | Pos | Neg | Neu |
|---|---|---|---|
| Pos | 0.28 | **0.52** | 0.20 |
| Neg | 0.08 | **0.73** | 0.19 |
| Neu | 0.12 | **0.57** | 0.31 |

**Spammer** (learned):

| | Pos | Neg | Neu |
|---|---|---|---|
| Pos | 0.35 | 0.31 | 0.34 |
| Neg | 0.32 | 0.36 | 0.32 |
| Neu | 0.30 | 0.37 | 0.33 |

**DS discovered:** the expert is reliable, the biased annotator overuses Neg, and the spammer is near-random — **all without us telling it who is who**.

# DS Results: Posterior vs. Majority Vote

**Contentious item:** Expert says Neg, Biased says Neg, Decent says Pos. **MV:** Neg (2 vs. 1).

**Learned matrices (from slide 33):**

| **Expert** | P | N | U |
|---|---|---|---|
| P | .91 | .05 | .04 |
| N | .03 | .93 | .04 |
| U | .05 | .03 | .92 |

| **Biased** | P | N | U |
|---|---|---|---|
| P | .28 | .52 | .20 |
| N | .08 | .73 | .19 |
| U | .12 | .57 | .31 |

| **Decent** | P | N | U |
|---|---|---|---|
| P | .76 | .13 | .11 |
| N | .10 | .79 | .11 |
| U | .12 | .10 | .78 |

**DS posterior** (same E-step as slide 28 — prior $\times$ likelihoods, normalize):

$$\text{Pos:} \quad \pi_P \times \underbrace{0.05}_{\text{Exp says N}} \times \underbrace{0.52}_{\text{Bias says N}} \times \underbrace{0.76}_{\text{Dec says P}} = 0.0089$$

$$\text{Neg:} \quad \pi_N \times \underbrace{0.93}_{\text{Exp says N}} \times \underbrace{0.73}_{\text{Bias says N}} \times \underbrace{0.10}_{\text{Dec says P}} = 0.0238$$

$$\text{Neu:} \quad \pi_U \times \underbrace{0.03}_{\text{Exp says N}} \times \underbrace{0.57}_{\text{Bias says N}} \times \underbrace{0.12}_{\text{Dec says P}} = 0.0004$$

Normalize: Pos = 27%, **Neg = 72%**, Neu = 1%. DS gives a **distribution**, not just a label.

# Experiment: Majority Vote vs. DS Accuracy

**We ran both methods on 80 items with known true labels:**

| Method | Accuracy | Items where methods disagree |
|---|---|---|
| Majority vote | 71.2% | 12 out of 80 items |
| DS (MAP label) | **78.8%** | |

**On the 12 items where they disagree, DS is correct 8 times, MV is correct 4 times.**

**Why?** DS down-weights the spammer and biased annotator. Majority vote counts their labels equally with the expert's.

**Takeaway**

DS helps most when annotators **differ in quality**. If all annotators were equally good, majority vote would be hard to beat.

# What Happens as We Change the Noise?

**We varied the spammer's noise level and measured accuracy:**

| Spammer diagonal | MV accuracy | DS accuracy | DS gain |
|---|---|---|---|
| 0.70 (decent) | 80.0% | 81.2% | +1.2% |
| 0.50 (weak) | 76.2% | 80.0% | +3.8% |
| 0.33 (random) | 71.2% | 78.8% | +7.6% |
| 0.10 (adversarial) | 63.8% | 76.2% | +12.4% |

**Pattern:** the worse the spammer, the more DS helps.

When annotators are similar quality (top row), MV is nearly as good.
When there's a bad annotator (bottom rows), MV gets dragged down but DS learns to ignore them.

**DS is not magic** — it needs enough data to learn the confusion matrices. With very few items, the estimates are noisy.

# From DS to Soft Labels

**DS gives a probability distribution for each item.** Example:

$$\text{Item 7:} \quad \text{Pos} = \mathbf{93\%}, \quad \text{Neg} = 4\%, \quad \text{Neu} = 2\%$$

**You can use this in two ways:**

| Approach | What you do | Example |
|---|---|---|
| **Hard label** | Pick the class with highest probability | "Pos" |
| **Soft label** | Use the full probability distribution | (0.93, 0.04, 0.02) |

**Why use soft labels?**

- They preserve **uncertainty**: a 51%/49% item is treated differently from a 99%/1% item
- The classifier learns "this item is ambiguous" rather than being forced into a hard choice
- Clean items still get near-100% targets, so they're not harmed

In practice, soft-label training uses the same loss function (cross-entropy) — you just replace the one-hot target with the DS distribution. It's a one-line code change.

# Hard vs. Soft Label Training

**Comparison: training a classifier on majority vote vs. DS posteriors.**

| Training labels | Test accuracy | Test cross-entropy |
|---|---|---|
| Majority vote (hard) | 74.5% | 0.82 |
| DS MAP (hard) | 77.2% | 0.71 |
| DS posterior (soft) | **78.8%** | **0.65** |

**Soft labels win on both metrics** because:

- Ambiguous items don't force the model into overconfident predictions
- The model calibration improves (cross-entropy goes down)
- Clean items still get near-one-hot targets, so they're not harmed

## Practical note

Soft-label training uses the same loss function (cross-entropy) — you just replace the one-hot target vector with the DS posterior vector. One line of code changes.

# Mixed Human + LLM Annotators

**From Lecture 11:** an LLM configuration is just another annotator.

**DS handles this naturally:**

- Treat each LLM configuration (model + prompt + temperature) as an annotator
- DS learns its confusion matrix just like any human annotator
- If the LLM is biased (e.g., overuses one label), DS detects it

**Example: 2 humans + 1 LLM on a sentiment task.**

| Annotator | Pos diag | Neg diag | Neu diag | Note |
|---|---|---|---|---|
| Human 1 | 0.85 | 0.82 | 0.70 | Struggles with Neutral |
| Human 2 | 0.80 | 0.88 | 0.75 | Better at Negative |
| LLM (Prompt A) | 0.90 | 0.85 | 0.50 | Poor on Neutral |

DS learns the LLM is strong on Positive/Negative but weak on Neutral — and weights its labels accordingly.

# LLMs with Logprobs: A Special Case

**If you have LLM logprobs** (from Lecture 11), you have two options:

**Option A: Hard labels into DS**

- Take the argmax of the LLM's distribution
- Feed it into DS as a regular annotator label
- DS learns the confusion matrix from the hard labels

**Option B: Use LLM probabilities directly as soft evidence**

- The LLM's probability distribution is already a "soft annotation"
- Combine it with DS posteriors from human annotators
- Use LLM probabilities as a feature or triage signal

**Option A is simpler and works well in practice.**
Option B is more principled but more complex to implement.

Remember from Lecture 11: LLM confidence can be miscalibrated. DS helps because it learns the LLM's actual error patterns, not just its self-reported confidence.

# LLM-Assisted Adjudication

**Emerging approach for 2025+**

**Options:**

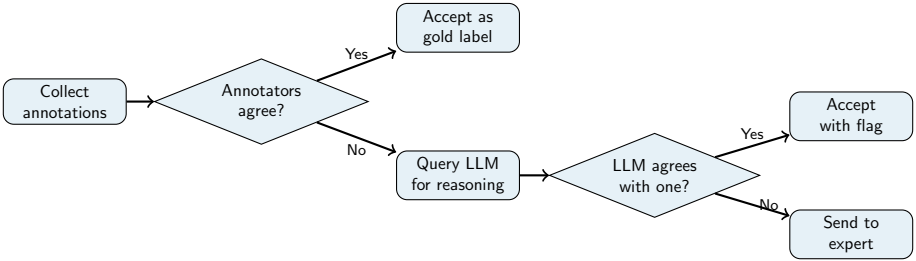
1. **LLM as tie-breaker:**
   - When annotators disagree, ask LLM
   - Use as third "vote"
2. **LLM reasoning:**
   - Ask LLM to explain which label is correct
   - Human reviews LLM reasoning
3. **LLM confidence:**
   - Accept human majority if LLM confidence low
   - Review if LLM disagrees with high confidence

**Caution**

LLM shouldn't be sole adjudicator for evaluation data — circular evaluation risk if the same model family is later tested on that data.

# LLM Adjudication in Practice

**A practical workflow:**




**Benefits:**

- Reduces expert review load by 40–60%
- LLM reasoning helps experts decide faster
- Creates audit trail of disagreement rationales

# When to Model vs. When to Just Vote

**Use DS / Bayesian DS when:**

- Annotators vary meaningfully in reliability or bias
- You have 2–5+ labels per item from different annotators
- You need soft targets for downstream training
- You want to **diagnose** annotator behavior (find spammers, biased workers)

**Majority vote is often fine when:**

- Annotators are similarly competent and errors are symmetric
- You have many annotators per item (noise cancels out)
- The task is objective enough that disagreement is rare

**DS may not help when:**

- You have 1 or fewer labels per item (not enough data for confusion matrices)
- Annotators are correlated (shared guidelines, same LLM prompt)
- The task is genuinely subjective ("true label" may not exist)

# DS Assumptions to Keep in Mind

**Three assumptions that can be violated:**

1. **Conditional independence**
   Annotators' errors are independent given the true label.
   Violated when: shared training, shared LLM prompts, crowdworker collusion.
2. **Stationary annotator behavior**
   Each annotator's confusion matrix is fixed over time.
   Violated when: LLM backend updates, annotator fatigue, evolving understanding.
3. **No item difficulty**
   DS assumes all items are equally hard — the only variation is annotator quality.
   Violated when: some items are genuinely ambiguous (sarcasm, edge cases).

**These are limitations, not deal-breakers.** DS is still a strong baseline.
Extensions exist for item difficulty, correlated annotators, and non-stationary behavior — but they're beyond this course.

# Creating Gold Standard Dataset

**Complete workflow:**

1. Multiple annotators label data
2. Calculate IAA to verify quality
3. Identify disagreements
4. Apply adjudication strategy
5. Verify adjudicated labels
6. Document process

**Best practices:**

- Keep original annotations (for analysis)
- Document adjudication decisions
- Track problematic patterns
- Update guidelines for future annotation

# Gold Standard Data Splits

**Splitting your gold standard for ML:**


**Critical rules:**

- **Test set:** Highest quality adjudication (expert review preferred)
- **Training set:** Majority voting acceptable (noise is tolerable)
- **Dev set:** Used for tuning — moderate quality needed
- **Never** let adjudication leakage cross split boundaries

**Stratify:** Ensure label distribution is balanced across splits

# Documenting Adjudication Decisions

**Every adjudication decision should be traceable:**

| Item | Labels | Strategy | Decision | Rationale |
|---|---|---|---|---|
| sent_042 | A: Pos, B: Neg | Expert review | Negative | Rhetorical structure signals negative |
| sent_107 | A: Neu, B: Pos, C: Pos | Majority vote | Positive | 2/3 agree |
| sent_215 | A: Neg, B: Neg, LLM: Pos | Human majority | Negative | LLM misread sarcasm |

**Why document?**

- Reproducibility: others can audit your gold standard
- Error analysis: patterns in disagreements reveal task difficulty
- Guideline improvement: recurring issues inform future revisions

# Discussion: Quality vs. Scale

**A common tension in annotation projects:**

| High Quality | High Scale |
|---|---|
| Expert adjudication on all items | Majority voting only |
| Small dataset (1,000 items) | Large dataset (50,000 items) |
| $\kappa = 0.85$ | $\kappa = 0.65$ |
| High confidence in labels | Some label noise |

**Questions to discuss:**

- Which would you prefer for training a model? For evaluation?
- Can you have both? What strategies would help?
- How does the task complexity affect this tradeoff?

# Key Takeaways

1. **Soft / distributional metrics** (JSD, soft $\kappa$) reveal what hard labels hide—and disagreement can be signal, not just noise
2. **Observed labels are noisy measurements**, not ground truth — model the noise rather than averaging it away
3. **Dawid–Skene** learns per-annotator confusion matrices and infers true labels via EM
4. **DS beats majority vote** when annotators differ in quality—especially with spammers or biased annotators
5. **Bayesian DS** adds priors to stabilize estimates when per-annotator data is sparse
6. **LLMs can assist adjudication** as tie-breakers or reasoning aids, but shouldn't be the sole judge for evaluation data
7. **Gold standard creation** requires careful adjudication, documentation, and stratified data splits
8. **Always report majority vote as a baseline** — it's the sanity check you should not skip

Questions?

Office Hours: Wednesdays 1-3pm, Volen 109

jinzhao@brandeis.edu