COSI-230B: Natural Language Annotation for Machine Learning

Lecture 13: Modeling Annotator Reliability and Label Noise (Part II)

Jin Zhao

Brandeis University

March 16, 2026

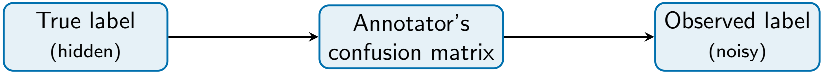
Today's Agenda

1. Continued from last time: EM walkthrough (computing posteriors)
2. The E-step and M-step in detail
3. DS EM results on toy data
4. Majority vote vs. DS under controlled noise
5. Soft labels for training + mixed human/LLM annotators
6. LLM-assisted adjudication
7. Limitations and decision rules
8. Creating gold standard datasets + documentation
9. Key takeaways

Recap: Why Dawid–Skene?

The problem: we learned to *measure* agreement (κ, soft κ, JSD), but **majority vote** treats every annotator equally — it fails when they differ in quality.

The DS idea: model each annotator's error pattern, then reverse-engineer the true labels.



The chicken-and-egg trick (EM):

1. Initialize with majority vote as a rough guess
2. If we knew true labels $\rightarrow$ we could learn confusion matrices
3. If we knew confusion matrices $\rightarrow$ we could infer true labels
4. Alternate steps 2–3 until convergence

Last time (Lecture 12)

We set up the DS model, defined confusion matrices, and began an EM walkthrough on Item 7.
Today: we finish that walkthrough and see DS in action.

Recap: The Setup for Item 7

Item 7: Expert says Pos, Biased says Neg, Spammer says Neg. **Majority vote:** Neg (2 vs. 1).

DS idea: try each possible true label, multiply prior $\times$ likelihoods from each annotator.

Confusion matrices (rows = true label, cols = annotator 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

Class priors (from MV initialization): 45% Pos, 35% Neg, 20% Neu

Today: we compute the score for each candidate true label and see which one wins.

EM Walkthrough: What If True = Pos?

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

Annotator	Said	Where to look (Lecture 12 confusion matrices)	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!)

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

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 almost *never* says Pos when the truth isn't Pos (only 4% of the time)
- So the Expert saying Pos is very strong evidence for true = Pos
- The Biased annotator says Neg regardless of the true label ($\geq$50% of the time) — not informative
- The Spammer is random (33% everywhere) — carries no information at all

DS weights votes by annotator reliability, not by headcount.

The E-Step: Posteriors for Every Item

What the EM walkthrough slides just showed 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	$P(\text{Pos})$	$P(\text{Neg})$	$P(\text{Neu})$
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	$P(t = P)$	Said P?
3	Neg	0.10	
5	Pos	0.80	✓
7	Pos	0.93	✓
12	Neg	0.05	
18	Pos	0.90	✓

Expert's matrix

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

Repeat for every cell, every annotator.

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

The M-Step: Another Cell (Biased Annotator)

Same method, different annotator. Compute one cell of the **Biased** annotator's matrix: $P(\text{Biased says Neg} \mid \text{true} = \text{Pos})$

(Same 5 items; same $P(\text{t} = \text{P})$ from E-step.)

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

Biased's matrix

	Says P	Says N	Says U
True P	?	**0.68**	?
True N	?	?	?
True U	?	?	?

The bias shows up:
says Neg 68% of the time

even when truth is Pos!

$$\underbrace{0.10 + 0.80 + 0.93 + 0.05}_{\text{said Neg, weighted}} = 1.88 \;\Big/\; \underbrace{0.10 + 0.80 + 0.93 + 0.05 + 0.90}_{\text{all items, weighted}} = 2.78 \quad = \quad \mathbf{0.68}$$

Compare: Expert's diagonal was **0.95**. The Biased annotator's off-diagonal is **0.68** — DS is learning who to trust.

The M-Step: Re-estimate Class Priors

The M-step also updates the class priors. Average the posteriors across items. (5 items shown; in practice, average over all 80.)

Item	$P(\text{Pos})$	$P(\text{Neg})$	$P(\text{Neu})$
3	0.10	0.85	0.05
5	0.80	0.15	0.05
7	0.93	0.04	0.02
12	0.05	0.90	0.05
18	0.90	0.03	0.07
Sum	**2.78**	**1.97**	**0.24**
Avg (÷5)	**0.56**	**0.39**	**0.05**

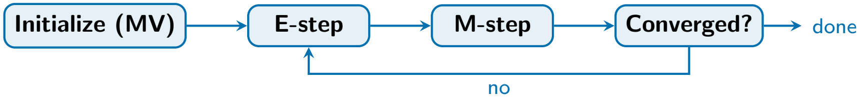
Compare to MV initialization: Pos = 0.45, Neg = 0.35, Neu = 0.20

What changed?

- **Pos went up** (0.45 → 0.56): DS shifted items *toward* Pos that MV had wrong
- **Neg similar** (0.35 → 0.39): most Neg items already correctly identified
- **Neu dropped** (0.20 → 0.05): items MV called Neutral were reassigned by DS

The Full EM Loop

Now we have updated priors and confusion matrices. What next? Loop back!



Each iteration improves both estimates:

- Better posteriors → better matrices → even better posteriors
- Typically **5–20 iterations**; first 2–3 do most of the work
- **Converged** = posteriors change by $< \epsilon$ (e.g. 10^{-4}) between iterations — the estimates have stabilized and further loops won't change the answer

Summary

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

Experiment Setup: Simulate a Synthetic Dataset

Why simulate? We need data where we *know* the true labels, so we can check if DS recovers them.

The recipe: we play the role of "nature" and generate labels step by step.

```
# Step 1: Pick true labels for 80 items
true_labels = np.random.choice(
    ["Pos", "Neg", "Neu"], size=80,
    p=[0.45, 0.35, 0.20])    # 45% Pos, 35% Neg, 20% Neu

# Step 2: For each item, pick 3 of 4 annotators
for item in range(80):
    annotators = np.random.choice(4, size=3, replace=False)

    # Step 3: Each annotator labels using their confusion matrix
    for ann in annotators:
        row = confusion_matrix[ann][true_labels[item]]
        observed_label = np.random.choice(
            ["Pos","Neg","Neu"], p=row)  # noisy!
```

Key point: DS only sees the observed labels (Step 3 output). It does **not** see the true labels (Step 1) or the confusion matrices (Step 3 input).

Experiment Setup: The Four Annotator Profiles

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

Reading a row: "If the true label is Pos, the Expert says Pos 92% of the time, Neg 4%, Neu 4%."

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

Running DS: The EM Code

Here is the complete EM loop — surprisingly short:

```
# Initialize: use majority vote as first guess for true labels
posteriors = majority_vote_one_hot(labels)  # shape: (80, 3)

for iteration in range(20):
    # M-step: re-estimate confusion matrices and priors
    for ann in range(4):
        for true_class in range(3):
            for said_class in range(3):
                # soft count: weight by posterior probability
                conf[ann][true_class][said_class] = (
                    sum of posteriors[i][true_class]
                    where annotator ann said said_class on item i
                ) / sum of posteriors[i][true_class] for all items i
    priors = posteriors.mean(axis=0)  # average across items

    # E-step: re-estimate posteriors for each item
    for item in range(80):
        for c in range(3):  # try each candidate true label
            score[c] = priors[c]
            for ann in annotators_of[item]:
                score[c] *= conf[ann][c][label[ann][item]]
        posteriors[item] = score / score.sum()  # normalize
```

The M-step updates matrices + priors (slides 9–11). The E-step computes posteriors (slides 5–8). That's it!

DS Results: Learned vs. True Confusion Matrices

After running EM, DS recovers the annotator profiles. Compare true (left) vs. learned (right):

Expert

	True			Learned		
	P	N	U	P	N	U
P	**.92**	.04	.04	**.91**	.05	.04
N	.04	**.92**	.04	.03	**.93**	.04
U	.04	.04	**.92**	.05	.03	**.92**

Decent

	True			Learned		
	P	N	U	P	N	U
P	**.78**	.11	.11	**.76**	.13	.11
N	.11	**.78**	.11	.10	**.79**	.11
U	.11	.11	**.78**	.12	.10	**.78**

Biased

	True			Learned		
	P	N	U	P	N	U
P	.30	**.50**	.20	.28	**.52**	.20
N	.10	**.70**	.20	.08	**.73**	.19
U	.10	**.60**	.30	.12	**.57**	.31

Spammer

	True			Learned		
	P	N	U	P	N	U
P	.33	.33	.33	.35	.31	.34
N	.33	.33	.33	.32	.36	.32
U	.33	.33	.33	.30	.37	.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 16):

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: prior × likelihoods, normalize. **Priors** (from M-step): $P(\text{P}) = 0.45$, $P(\text{N}) = 0.35$, $P(\text{U}) = 0.20$

$$\text{Pos:}\quad \underbrace{0.45}_{\text{prior}} \times \underbrace{0.05}_{\text{Exp says N}} \times \underbrace{0.52}_{\text{Bias says N}} \times \underbrace{0.76}_{\text{Dec says P}} = 0.0089$$

$$\text{Neg:}\quad \underbrace{0.35}_{\text{prior}} \times \underbrace{0.93}_{\text{Exp says N}} \times \underbrace{0.73}_{\text{Bias says N}} \times \underbrace{0.10}_{\text{Dec says P}} = 0.0238$$

$$\text{Neu:}\quad \underbrace{0.20}_{\text{prior}} \times \underbrace{0.03}_{\text{Exp says N}} \times \underbrace{0.57}_{\text{Bias says N}} \times \underbrace{0.12}_{\text{Dec says P}} = 0.0004$$

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.

Soft Labels in Code: The One-Line Change

Hard labels (majority vote or DS MAP) — standard training:

```
# Hard label: one-hot vector
target = [0, 1, 0]                    # "Neg" with 100% confidence
loss = cross_entropy(model_output, target)
```

Soft labels (DS posteriors) — the only change is the target:

```
# Soft label: DS posterior distribution
target = [0.27, 0.72, 0.01]    # "probably Neg, maybe Pos"
loss = cross_entropy(model_output, target)  # same function!
```

That's it. The loss function is identical — `cross_entropy` already accepts any probability distribution as the target. No architecture changes, no new hyperparameters.

The effect: the model learns that this item is *probably* Neg but has some uncertainty, rather than being told it's *definitely* Neg.

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.

LLM-Assisted Adjudication

Emerging approach for 2025+

Options:

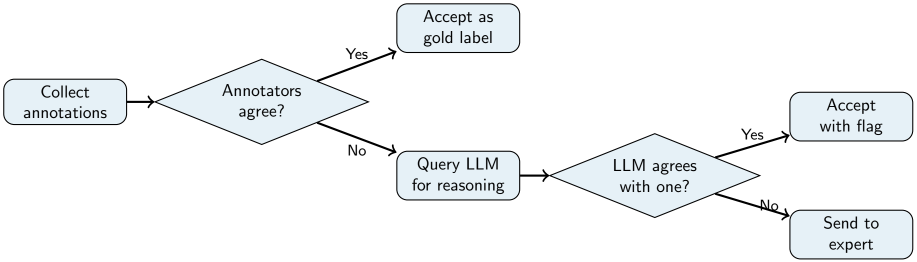
1. **LLM as tie-breaker:**
   - When annotators disagree, ask LLM
   - Use as third "vote"
2. **LLM reasoning:**
   - Ask LLM to explain which label is correct
   - Human reviews LLM reasoning
3. **LLM confidence:**
   - Accept human majority if LLM confidence low
   - Review if LLM disagrees with high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

LLM Adjudication: A Worked Example

Sentiment task. Two annotators disagree on a tweet:

Tweet: "The new update *finally* fixed the battery issue. Only took them six months."

Annotator A: Positive **Annotator B:** Negative

Step 1: Query the LLM with the text + both labels + guidelines:

```
prompt = """Two annotators disagree on this tweet's sentiment.
Tweet: "The new update finally fixed the battery issue.
       Only took them six months."
Annotator A: Positive    Annotator B: Negative
Which label is correct? Explain your reasoning in 2-3 sentences."""
```

Step 2: LLM response:

"The tweet is **Negative**. While it acknowledges the fix (positive surface), the phrase 'only took them six months' is sarcastic, expressing frustration with the slow response. The dominant sentiment is dissatisfaction."

Step 3: Human reviewer reads LLM reasoning, agrees → label = **Negative**, flagged as LLM-assisted.

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:

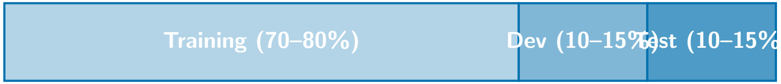
1. Multiple annotators label data
2. Calculate IAA to verify quality
3. Identify disagreements
4. Apply adjudication strategy
5. Verify adjudicated labels
6. Document process

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. **The EM walkthrough** shows how DS computes posteriors by multiplying priors and likelihoods, then normalizing
2. **DS beats majority vote** when annotators differ in quality—especially with spammers or biased annotators
3. **Soft labels from DS** preserve uncertainty and improve both accuracy and calibration in downstream training
4. **Mixed human + LLM annotation** is handled naturally by DS—each source gets its own learned confusion matrix
5. **LLMs can assist adjudication** as tie-breakers or reasoning aids, but shouldn't be the sole judge for evaluation data
6. **Gold standard creation** requires careful adjudication, documentation, and stratified data splits
7. **Always report majority vote as a baseline** — it's the sanity check you should not skip

Questions?

Questions?

Office Hours: Wednesdays 1-3pm, Volen 109

✉ jinzhao@brandeis.edu