COSI-230B: Natural Language Annotation for Machine Learning

Lecture 14: From Annotations to Models

Jin Zhao

Brandeis University

Spring 2026

Today's Agenda

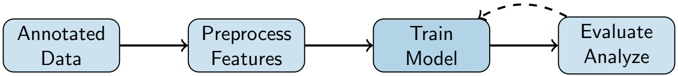
Part I: Training with Annotations

1. The modeling pipeline
2. Standard training approach
3. Handling annotation uncertainty
4. Soft labels and training
5. Multi-annotator learning
6. Human vs. LLM annotations

Part II: Evaluation

7. Evaluation metrics: P/R/F1
8. Multi-class metrics
9. Error analysis
10. Connecting errors to annotation quality
11. LLM-as-judge
12. Discussion and takeaways

The Modeling Pipeline



Core question: How does annotation quality affect each stage?

- Annotation choices propagate through the entire pipeline
- Evaluation results feed back into annotation improvement
- Today: from training decisions to evaluation methodology

Standard Training Approach

Using adjudicated gold standard:

1. Take gold label for each instance
2. Split into train/dev/test
3. Train model to predict labels
4. Evaluate on test set

Assumes:

- Single correct label exists
- Gold standard is reliable
- All instances equally informative

But: What about annotation uncertainty?

Handling Annotation Uncertainty

Disagreement carries information

Options:

1. **Ignore uncertainty:** Use gold labels only
2. **Filter uncertain:** Remove low-agreement items
3. **Weight by agreement:** Confident examples matter more
4. **Soft labels:** Train on label distributions
5. **Multi-task:** Model uncertainty explicitly

Key insight: Uncertain examples may be legitimately ambiguous—forcing a single label loses information

Soft Labels

Instead of single label, use distribution

Hard label:

- "This is POSITIVE" (one-hot: [1, 0, 0])

Soft label:

- "60% POSITIVE, 30% NEUTRAL, 10% NEGATIVE"
- Label distribution: [0.6, 0.3, 0.1]

From annotator votes:

- 3 annotators: 2 say Positive, 1 says Neutral
- Soft label: [0.67, 0.33, 0]

Training with Soft Labels

Standard cross-entropy loss:

$$L = -\sum_c y_c \log(p_c)$$

Where y is one-hot (hard label)

With soft labels:

$$L = -\sum_c \hat{y}_c \log(p_c)$$

Where $\hat{y}$ is the label distribution

Effect:

- Model learns nuance in uncertain cases
- Less confident predictions for ambiguous items
- Can improve generalization

Multi-Annotator Learning Approaches

Don't aggregate—model all annotators

Approaches:

1. **Data augmentation:** Treat each annotation as separate training example
2. **Multi-task learning:** Predict each annotator's label
3. **Annotator modeling:** Learn annotator-specific biases
4. **Ensemble:** Train separate models, combine predictions

Data augmentation example:

- Text: "Not bad at all" with annotations [Pos, Pos, Neu]
- Creates 3 training examples—model sees all perspectives

Benefits: Captures systematic disagreement, models perspective diversity, better for subjective tasks

Human vs. LLM Annotations for Training

Comparing annotation sources

Experiment design:

1. Annotate same data with humans AND LLM
2. Train separate models on each
3. Evaluate both on human-annotated test set
4. Compare performance

Key findings (various studies):

- LLM-trained models often comparable for simple, objective tasks
- Human annotations better for subjective/complex tasks
- Combined often best—LLM for volume, humans for quality

Remember: Always evaluate on human-annotated test data

Evaluation Metrics: Precision, Recall, F1

For classification:

Precision:

$$P = \frac{TP}{TP + FP}$$

How many predictions are correct?

Recall:

$$R = \frac{TP}{TP + FN}$$

How many actual positives found?

F1 Score:

$$F_1 = \frac{2 \cdot P \cdot R}{P + R}$$

Harmonic mean of precision and recall

Multi-Class Metrics

How to aggregate across classes:

Macro-averaged:

- Calculate metric for each class, average with equal weight
- Good for imbalanced data—treats rare classes equally

Micro-averaged:

- Pool all predictions, calculate single metric
- Dominated by frequent classes

Weighted:

- Weight by class frequency
- Balance between macro and micro

Report what makes sense for your task—often multiple

Error Analysis

Beyond aggregate metrics:

Confusion matrix:

- Which classes are confused with each other?
- Systematic patterns in errors?

Error categorization:

- Boundary errors (span too long/short)
- Type errors (wrong category)
- Missing errors (entity not found)
- Spurious errors (non-entity labeled)

Best practice: Report per-class breakdown, confidence intervals, and common failure modes alongside aggregate metrics

Connecting Errors to Annotation Quality

Key questions:

1. **Do model errors correlate with low IAA?**
   - Categories with low agreement $\rightarrow$ lower model performance
   - Ambiguous items are hard for both humans and models
2. **Which guidelines need improvement?**
   - Systematic errors point to unclear definitions
   - Confusion between specific categories suggests overlap
3. **Feedback loop:**
   - Model errors $\rightarrow$ refine guidelines $\rightarrow$ re-annotate $\rightarrow$ retrain
   - Annotation quality sets the ceiling for model performance

Reporting Results

What to include:

1. **Metrics:** P, R, F1 (macro and micro)
2. **Confidence intervals:** Bootstrap or cross-validation
3. **Baselines:** For comparison (majority class, simple rules)
4. **Per-class breakdown:** Identify weak points
5. **Error analysis:** Common failure modes
6. **Statistical significance:** If comparing systems

Example: "Our model achieves 78.3 F1 ($\pm$1.2) on entity-level evaluation, compared to 72.1 F1 for the baseline. Performance on PER (85.2) exceeds ORG (71.4)."

Common Evaluation Pitfalls

Avoid these mistakes:

1. **Test set contamination:** Training on test data (or LLM memorizing it)
2. **Overfitting to dev:** Too much tuning on dev set
3. **Cherry-picking metrics:** Only reporting best metric
4. **Missing baselines:** No comparison point
5. **Ignoring variance:** Single run without confidence intervals
6. **Unfair comparisons:** Different preprocessing or data splits

For LLM evaluation: Also check for benchmark contamination—the test set may be in the LLM's training data

Building Good Evaluation Benchmarks

Requirements:

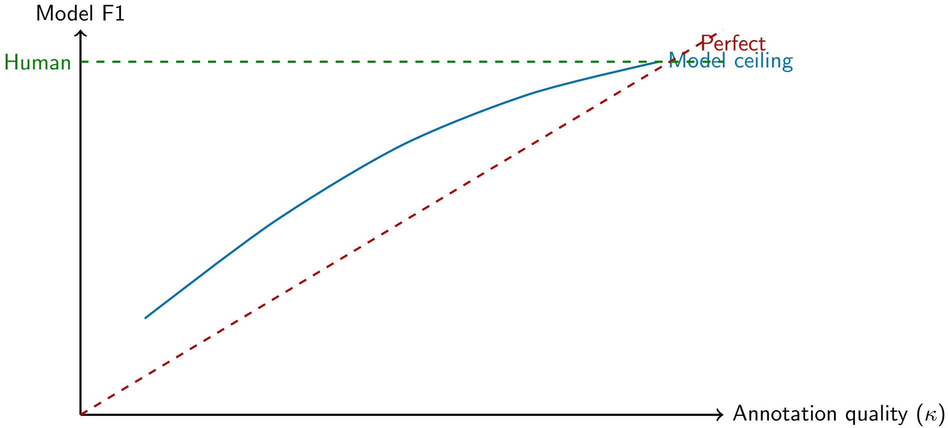
1. High-quality human annotations (adjudicated)
2. Representative of the real task distribution
3. Held out from all training
4. Well-documented (guidelines, annotator info, IAA)

Decontamination for LLMs:

- Ensure test data not in LLM training corpus
- Use recent data after training cutoff
- Check for n-gram overlap

Anti-patterns: LLM-generated test labels, too-small samples, undocumented preprocessing

Annotation Quality and Model Ceilings



Key insight: Model performance is bounded by annotation quality.
Improving annotations from $\kappa = 0.6$ to $\kappa = 0.8$ often helps more than switching from BERT to GPT-4

LLM-as-Judge

Using LLMs to evaluate model outputs

Setup:

1. Model generates output
2. LLM evaluates quality (1–5 scale, pass/fail, etc.)
3. Aggregate LLM scores

Advantages:

- Scalable to large test sets
- Consistent (unlike variable human judges)
- Can provide explanations for scores

Disadvantages:

- Bias toward LLM-style preferences
- May not match human judgment
- Circular if evaluating LLMs with LLMs

LLM-as-Judge Cautions

When NOT to use:

- Evaluating the same LLM that is judging
- Tasks where LLMs are known to fail
- High-stakes decisions
- Creating research benchmarks

Best practices:

- Validate against human judgments first
- Report correlation with human evaluators
- Use as supplement, not replacement
- Be transparent about methodology

Bottom line: LLM-as-judge is a tool, not a shortcut—always ground in human evaluation

Discussion: When Does Annotation Quality Limit Model Performance?

Consider these scenarios:

1. Your model achieves 85% F1 but IAA is only 80%—is the model "good enough"?
2. Two categories are frequently confused by both annotators and the model—what do you do?
3. LLM annotations give you 10x more training data but 5% lower quality—worth it?
4. Your error analysis shows the model struggles on the same items annotators disagree on—next steps?

Principle: Model performance is bounded by annotation quality—improving annotations often matters more than improving models

Key Takeaways

1. **Standard training** uses gold labels but ignores uncertainty
2. **Soft labels** capture annotation distributions for nuanced training
3. **Multi-annotator learning** models all perspectives, not just majority
4. **Human vs. LLM annotations:** choose based on task complexity
5. **P/R/F1** are standard metrics—know macro vs. micro vs. weighted
6. **Error analysis** should connect model failures to annotation quality
7. **LLM-as-judge** is useful but requires human validation
8. **Annotation quality** sets the ceiling for model performance

Questions?

Questions?

Office Hours: Wednesdays 1–3pm, Volen 109

jinzhao@brandeis.edu