COSI-230B: Natural Language Annotation for Machine Learning

Lecture 21: Active Learning and LLMs as Annotators-in-the-Loop

Jin Zhao

Brandeis University

April 22, 2026

Today's Agenda

Part I: Active Learning Foundations

- Active learning recap
- Classic acquisition functions
- The cold-start problem

Part II: LLMs as Annotators

- Are LLMs good annotators?
- GPT-4 domain analysis
- LLMaAA: LLMs as active annotators
- ActiveLLM: LLMs as selection engines

Part III: Risks & Routing

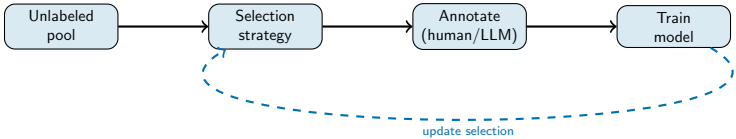
- Human vs. LLM routing
- Cost-quality tradeoff analysis
- Selection bias and coverage gaps
- Feedback loops and self-confirmation

Part IV: Practice & Wrap-Up

- Code demos
- In-class activity
- Key takeaways and readings

Goal: Understand how LLMs change both *what gets selected* and *who labels it*—and the resulting risks.

Active Learning in One Minute



Core idea: Label the *most informative* examples, not random ones.

Why it matters for annotation:

- Annotation is expensive—active learning reduces the number of labels needed
- Typical savings: 30–70% fewer labels for equivalent model performance
- But: you are choosing a *non-random* training set—this has consequences

> *"Active learning is power: you control what the model learns by controlling what gets labeled."*

Classic Acquisition Functions

How do we decide which examples to label next?

1. **Uncertainty Sampling** — Label examples the model is least confident about:

$$x^* = \arg\max_{x \in \mathcal{U}} \; H\big(P(y \mid x; \theta)\big) \quad \text{where } H(p) = -\sum_c p_c \log p_c$$

2. **Query-by-Committee (QBC)** — Train a committee of K models; select where they disagree:

$$x^* = \arg\max_{x \in \mathcal{U}} \; \frac{1}{K} \sum_{k=1}^{K} \mathrm{KL}\big(P_k(y|x) \parallel P_{\text{avg}}(y|x)\big)$$

Classic Acquisition Functions (cont.)

3. **Expected Model Change** — Select the example that would cause the largest gradient update:

$$x^* = \arg\max_{x \in \mathcal{U}} \mathbb{E}_y \left[\|\nabla_\theta \, \ell(y, f_\theta(x))\| \right]$$

4. **Diversity Sampling** — Select examples that maximize coverage of the feature space (e.g., k-medoids on embeddings)

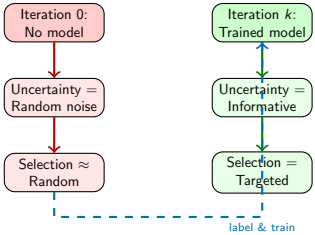
Settles (2012). Active Learning. Synthesis Lectures on Artificial Intelligence and Machine Learning.
https://www.morganclaypool.com/doi/abs/10.2200/S00429ED1V01Y201207AIM018

The Cold-Start Problem

Problem: At iteration 0, there is no trained model $\Rightarrow$ uncertainty estimates are **meaningless**.

Why?

- Random initialization $\rightarrow$ random uncertainty
- Model hasn't seen any task-relevant data
- All examples look equally "uncertain"
- Selection degenerates to random sampling



Common workarounds:

- Seed with a small random sample (10–50 examples)
- Use pre-trained embeddings for diversity selection

Are LLMs Good Annotators?

Mohta et al. (2023); Bhat et al. (2023): Systematic evaluation of LLMs vs. human annotators.

Task Type	LLM Accuracy	Human Accuracy	Winner
Sentiment (binary)	89–93%	85–90%	LLM ≈ Human
NLI (3-class)	82–87%	88–92%	Human (slight edge)
Named Entity Recognition	78–83%	92–96%	Human
Toxicity Detection	80–86%	82–88%	Comparable
Subjective/Cultural	60–72%	80–90%	Human (large gap)
Paraphrase Detection	85–90%	88–92%	LLM ≈ Human
Complex Reasoning	65–75%	85–92%	Human

Mohta et al. (2023). ICML Workshop. Bhat et al. (2023). Eval4NLP Workshop.

`https://proceedings.mlr.press/v239/mohta23a/mohta23a.pdf`

Are LLMs Good Annotators? (cont.)

Key Finding

LLMs are competitive on **well-defined, objective tasks** but fall behind on tasks requiring **world knowledge, cultural context, or nuanced judgment**.

LLMs as Annotators: Empirical Evaluations and Caveats

The practical picture:

Evidence for:

- Competitive with crowd-workers on many tasks
- Consistent (no fatigue/drift)
- Fast and cheap at scale
- Good at well-defined tasks (sentiment, NLI)
- Reproducible—same input $\rightarrow$ same label (at temp=0)

Evidence against:

- Systematic biases (not random noise)
- Struggle with subjective/cultural tasks
- Training on LLM labels can *underperform* human labels
- Self-confirmation risk in active loops
- Label distribution may not match true prior

Default Caution

Training on LLM-generated labels can underperform human-labeled training—treat this as a default caution, not a blanket ban. Always validate with a human-labeled test set.

GPT-4 as Annotator: Domain Analysis

Where does GPT-4 excel, and where does it fail as an annotator?

Excels:

- **Sentiment analysis:** Clear positive/negative signals
- **Grammaticality judgments:** Strong language model prior
- **Topic classification:** Well-separated categories
- **Textual entailment:** Logical reasoning tasks
- **Spam/toxicity:** Pattern recognition tasks

Fails:

- **Sarcasm/irony:** Cultural and contextual nuance
- **Hate speech:** Evolving norms, community-specific language
- **Emotion (fine-grained):** Subjective, annotator-dependent
- **Domain-specific NER:** Biomedical, legal entities
- **Discourse relations:** Complex pragmatic reasoning

GPT-4 as Annotator: Domain Analysis (cont.)

Rule of Thumb

If the task has a clear right answer that educated English speakers would agree on, GPT-4 is likely a good annotator. If the task requires community norms, domain expertise, or subjective judgment, prefer humans.

LLMaAA: LLMs as Active Annotators

Zhang et al. (2023): Making Large Language Models as Active Annotators.

Design

1. LLM annotates a batch of examples (pseudo-labels)
2. Train a smaller task model on pseudo-labels
3. Use task model's uncertainty to select next batch
4. LLM annotates selected batch
5. Repeat with reweighting to handle label noise

Key innovation: Combines active selection with LLM pseudo-labeling, plus reweighting to learn robustly from noisy labels.

Risk: The active loop can converge to regions where the LLM is confident but wrong—self-confirmation trap.

Zhang et al. (2023). LLMaAA. Findings of EMNLP 2023. https://aclanthology.org/2023.findings-emnlp.872/

LLMaAA: Full Algorithm & Reweighting

Algorithm:

1. **Init:** Sample seed set $\mathcal{S}_0 \subset \mathcal{U}$ randomly
2. **Annotate:** $\tilde{y}_i = \mathrm{LLM}(x_i)$ for $x_i \in \mathcal{S}_0$
3. **For** $t = 1, 2, \ldots, T$:
   (a) Train f_θ on $\{(x_i, \tilde{y}_i, w_i)\}$
   (b) Compute uncertainty: $u(x) = H(P(y|x; \theta))$
   (c) Select: $\mathcal{S}_t = \text{top-}k$ by $u(x)$
   (d) Annotate: $\tilde{y}_i = \mathrm{LLM}(x_i)$, $x_i \in \mathcal{S}_t$
   (e) Add to labeled set with weights

Reweighting mechanism:

$$w_i = \frac{P_\theta(\tilde{y}_i \mid x_i)^\beta}{\sum_j P_\theta(\tilde{y}_j \mid x_j)^\beta}$$

- $\beta > 0$: upweights examples where the model agrees with the LLM label
- Intuition: if model and LLM disagree, the pseudo-label may be wrong $\rightarrow$ downweight
- Prevents noisy labels from dominating training

Results (SST-2):

LLMaAA: 91.8% (200 labels)

Random+LLM: 89.2%

ActiveLLM: LLMs as Selection Engines

Bayer & Reuter (2024/2025): Use LLMs to select informative data points in few-shot scenarios.

Key idea: LLMs can help pick items *before* any task model exists—addressing the **cold-start problem**.

Traditional active learning:

- Needs a trained model for uncertainty
- Cold-start: random selection initially
- Model-dependent selection

ActiveLLM:

- LLM provides uncertainty estimates
- No cold-start: works from iteration 0
- Model-agnostic selection

Limitation: LLM's uncertainty may not match the task model's uncertainty—selection may be suboptimal for the downstream model.

Bayer & Reuter (2024/2025). ActiveLLM. arXiv:2405.10808.

ActiveLLM: LLM Uncertainty vs. Task Model Uncertainty

How do LLM uncertainty estimates compare to task model uncertainty?

LLM Uncertainty Sources:

- **Token-level entropy:** $H(P(\text{token}|x))$ from next-token probabilities
- **Self-consistency:** Variance across multiple samples at $T > 0$
- **Verbalized confidence:** "I am 70% sure this is positive"
- **Prompt sensitivity:** Different prompts $\rightarrow$ different labels

Key Findings:

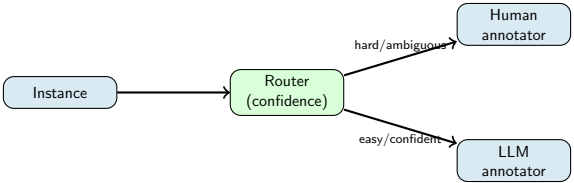
- LLM uncertainty and task model uncertainty have **moderate correlation** ($\rho \approx 0.3$–0.5)
- LLM is uncertain about **different** examples than a small classifier
- Best results: **combine** both signals
- LLM uncertainty is most useful in **early** AL iterations

Practical Implication

Use LLM uncertainty for cold-start (iterations 0–2), then transition to task model uncertainty as labeled data grows. Hybrid strategies outperform either alone.

Human vs. LLM Routing: When to Pay for Humans

Rouzegar et al. (2024): Route instances to human or LLM annotator based on cost-quality tradeoff.



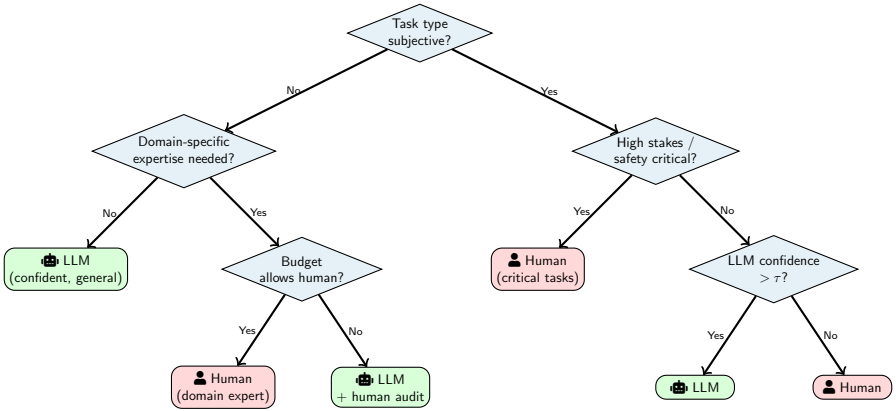
Decision criteria:

- **LLM confidence:** Route low-confidence items to humans
- **Task type:** Subjective/cultural tasks → humans; factual/structured → LLM
- **Budget:** Allocate human effort where it adds most value

Rouzegar et al. (2024). Enhancing Text Classification through LLM-Driven Active Learning.

`https://aclanthology.org/2024.law-1.10.pdf`

Decision Tree: Human vs. LLM Annotation



The threshold τ is task-dependent; calibrate on a small human-labeled validation set.

Cost-Quality Tradeoff Analysis

Annotation Method	Cost / Instance	Typical Accuracy	Best For
Expert human annotator	\$0.50–\$5.00	92–98%	Complex, high-stakes tasks
Crowd-worker (MTurk)	\$0.05–\$0.50	75–90%	Simple, scalable tasks
GPT-4 (API)	\$0.01–\$0.10	80–93%	Well-defined, objective tasks
GPT-3.5 (API)	\$0.001–\$0.01	70–85%	Bulk pseudo-labeling
Open-source LLM (local)	$\sim$\$0.001	65–80%	Privacy-sensitive, budget tasks
LLM + human audit (10%)	\$0.05–\$0.60	85–95%	Balanced cost/quality
Active + LLM (LLMaAA)	\$0.005–\$0.05	85–92%	Label-efficient scenarios

Budget Planning Formula

$$\text{Total cost} = n_{\text{human}} \times c_{\text{human}} + n_{\text{LLM}} \times c_{\text{LLM}} + n_{\text{audit}} \times c_{\text{audit}}$$

Active learning reduces n_{total}; routing optimizes the split between n_{human} and n_{LLM}.

Risk: Selection Bias and Coverage Gaps

The Problem

Active learning selects non-randomly → your training distribution is *not* your population distribution. Models "learn what you show them."

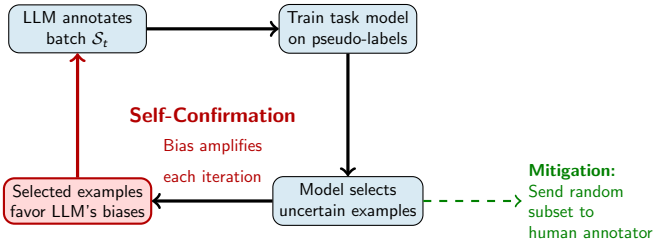
Selection bias manifests as:

- **Over-represented decision boundaries:** Dense annotation near uncertainty, sparse elsewhere
- **Missing subgroups:** Entire populations may be under-selected
- **Feedback loops:** Model's errors drive selection, creating self-reinforcing blind spots
- **Evaluation contamination:** Using the same LLM for selection *and* evaluation

Worst case: A classifier trained via active learning performs well on its selected distribution but fails catastrophically on deployment distribution.

Feedback Loops and Self-Confirmation

The danger cycle: When the LLM is both annotator and (indirect) selector.



Example

If GPT-4 systematically labels sarcastic reviews as "positive," the task model learns this pattern, becomes uncertain about *other* patterns, and never surfaces the sarcasm errors for correction.

Mitigations for Active Learning with LLMs

1. **Uncertainty + diversity:** Combine uncertainty sampling with diversity constraints to avoid coverage gaps
2. **Stratified pools:** Ensure selection covers all known subgroups/categories
3. **Holdout audits:** Evaluate on a *randomly sampled* holdout, not the actively selected distribution
4. **Human-in-the-loop gates:** Periodically send random samples to human annotators for calibration
5. **Separate evaluation LLM:** Never use the same LLM for both selection/annotation and evaluation
6. **Periodic random injection:** At each AL iteration, mix in 10–20% randomly selected examples alongside uncertainty-selected ones

Mitigations for Active Learning with LLMs (cont.)

Key Definitions

Cold start: Active learning needs signal before uncertainty is meaningful.
Pseudo-label: Label produced by a model; useful but risky when fed back.
Human/LLM routing: Decision policy for which instances go to which labeler.

Demo 1: Uncertainty Sampling vs. Random

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

rng = np.random.default_rng(0)
X = rng.normal(size=(1500, 10))
y = (X[:,0] + 0.3*X[:,1] > 0).astype(int)

pool_X, pool_y = X[:1200], y[:1200]
test_X, test_y = X[1200:], y[1200:]

def run(active=True, steps=10, batch=20, seed=0):
    rng = np.random.default_rng(seed)
    labeled = list(rng.choice(len(pool_X), size=20, replace=False))
    unlabeled = [i for i in range(len(pool_X)) if i not in labeled]
    accs = []
    for _ in range(steps):
        clf = LogisticRegression(max_iter=2000).fit(pool_X[labeled], pool_y[labeled])
        accs.append(accuracy_score(test_y, clf.predict(test_X)))
        if active:
            probs = clf.predict_proba(pool_X[unlabeled])[:,1]
            pick = np.argsort(np.abs(probs-0.5))[:batch]  # most uncertain
        else:
            pick = rng.choice(len(unlabeled), size=batch, replace=False)
        new = [unlabeled[i] for i in (pick.tolist() if active else pick.tolist())]
        labeled += new
        unlabeled = [i for i in unlabeled if i not in new]
    return accs

print("active␣␣:", run(True))
```

Demo 2: Selection Bias Visualization

Goal: Show how actively selected samples differ from the full distribution.

```
import numpy as np, matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression

rng = np.random.default_rng(42)
X = rng.normal(size=(1000, 2))
y = (X[:,0] + 0.5*X[:,1] > 0).astype(int)

# Seed with 20 random, then run 5 rounds of uncertainty sampling
labeled = list(rng.choice(1000, 20, replace=False))
unlabeled = [i for i in range(1000) if i not in labeled]
for _ in range(5):
    clf = LogisticRegression().fit(X[labeled], y[labeled])
    probs = clf.predict_proba(X[unlabeled])[:,1]
    pick = np.argsort(np.abs(probs - 0.5))[:30]
    labeled += [unlabeled[i] for i in pick]
    unlabeled = [i for i in range(1000) if i not in labeled]

fig, axes = plt.subplots(1, 2, figsize=(10,4))
axes[0].scatter(X[:,0], X[:,1], c=y, alpha=0.3, s=10)
axes[0].set_title("Full␣Distribution␣(1000␣points)")
axes[1].scatter(X[labeled,0], X[labeled,1], c=y[labeled], alpha=0.6, s=15)
axes[1].set_title(f"Actively␣Selected␣({len(labeled)}␣points)")
for ax in axes: ax.set_xlabel("x1"); ax.set_ylabel("x2")
plt.tight_layout(); plt.savefig("selection_bias.png", dpi=150)
```

Observation: Selected points cluster near the decision boundary, leaving large regions unsampled.

Discussion Questions

1. If pseudo-labels are cheaper but noisier, how do you decide when to switch from LLM to human annotation? What metrics would trigger the switch?
2. What's the worst-case selection bias scenario for deploying a classifier trained via active learning? Give a concrete example.
3. Should you ever evaluate on a test set created using the same LLM that labeled your training set? Why or why not?
4. Consider a hate speech detection system: would you use LLM annotation, human annotation, or a hybrid? How would you design the routing policy?

In-Class Activity: Design an Annotation Strategy (15 min)

Scenario

You have an **unlabeled pool of 1,000 sentences** for **sentiment classification** (positive/negative/neutral). Your **budget is $100**. Design an active learning + LLM annotation strategy.

Costs:

- Expert human annotator: $1.00 per sentence
- Crowd-worker (MTurk): $0.15 per sentence
- GPT-4 API: $0.03 per sentence
- GPT-3.5 API: $0.003 per sentence

In-Class Activity: Design an Annotation Strategy (cont.)

Your group should decide:

1. How many sentences will you label in total?
2. What is your cold-start strategy (first batch)?
3. What acquisition function will you use?
4. How will you split between human and LLM annotation?
5. How will you handle quality control / auditing?
6. What is your expected final model accuracy?

Deliverable: A one-paragraph strategy summary per group. Be ready to present in 2 minutes.

Example Strategy (Reveal After Activity)

One possible approach with $100 budget:

1. **Cold start:** Label 50 sentences with GPT-4 ($1.50) + 10 with expert human for calibration ($10.00)
2. **Active loop (5 iterations):**
   - Train classifier on labeled data
   - Select 100 most uncertain sentences per iteration
   - Route: top-20% uncertain $\rightarrow$ crowd-worker ($0.15 each), rest $\rightarrow$ GPT-4 ($0.03 each)
   - Cost per iteration: $20 \times \$0.15 + 80 \times \$0.03 = \$5.40$
   - 5 iterations: $27.00
3. **Quality audit:** Random 5% of GPT-4 labels to expert human: $\sim$25 sentences $\times$ $1.00 = $25.00
4. **Total:** $1.50 + $10.00 + $27.00 + $25.00 = $63.50 (with $36.50 buffer)
5. **Result:** 560 labeled sentences, expected accuracy: 88–91%

Key Takeaways

1. Active learning is itself an annotation choice—your selected dataset becomes your training distribution
2. Classic acquisition functions (uncertainty, QBC, diversity) each have strengths; combining them is usually best
3. The cold-start problem is real—LLMs can help bootstrap, but introduce their own biases
4. LLMs as annotators: competitive on well-defined tasks, but systematically biased on subjective ones
5. LLMaAA combines active selection with LLM labeling; reweighting helps but doesn't eliminate noise
6. Human/LLM routing is a cost-quality tradeoff requiring explicit policy decisions
7. Self-confirmation loops are the biggest risk—always inject randomness and human audits
8. Most important: your selected dataset is not your population—always evaluate on random holdouts

Readings

Required Readings:

- Zhang et al. (2023). *LLMaAA: Making Large Language Models as Active Annotators.* Findings of EMNLP.
- Bayer & Reuter (2024). *ActiveLLM: Large Language Model-based Active Learning for Textual Few-Shot Scenarios.* arXiv:2405.10808.

Recommended Readings:

- Mohta et al. (2023). *Are Large Language Models Good Annotators?* ICML Workshop.
- Bhat et al. (2023). *LLMs As Annotators.* Eval4NLP Workshop.
- Rouzegar et al. (2024). *Enhancing Text Classification through LLM-Driven Active Learning and Annotation.*
- Settles (2012). *Active Learning.* Synthesis Lectures on AI and ML. (Ch. 1–3)
- Topic: **Annotation Quality, Agreement, and Adjudication**
- Inter-annotator agreement metrics, adjudication strategies, and quality pipelines