COSI-230B: Natural Language Annotation for Machine Learning

Lecture 16: LLM-Based Annotation

Jin Zhao

Brandeis University

Spring 2026

Today's Agenda

1. The case for LLM annotation
2. Zero-shot annotation
3. Few-shot in-context learning
4. Prompt design patterns and best practices
5. Model selection
6. Structured output formats
7. Cost estimation and optimization
8. Quality assurance and hybrid approaches

The Case for LLM Annotation

Why consider LLMs for annotation?

Human annotation:

- \$0.10–\$1.00 per annotation
- Hours to days for batches
- Training required
- Quality varies
- Limited availability

LLM annotation:

- \$0.001–\$0.01 per annotation
- Seconds to minutes
- No training needed
- Consistent (for better or worse)
- Always available

Key finding (Gilardi et al., 2023): GPT-4 outperforms crowd-workers on many text annotation tasks

When LLM Annotation Works

Good candidates for LLM annotation:

- **Clear, objective criteria:** Tasks with well-defined categories
- **Similar to training data:** Tasks LLMs have "seen" before
- **High-resource languages:** English, Chinese, Spanish, etc.
- **Speed over perfection:** Preliminary labeling, bootstrapping
- **Scale requirements:** Large datasets where human annotation is prohibitive

Examples: Sentiment classification, topic labeling, NER for common entities, spam detection, language identification

When Human Annotation is Essential

LLMs struggle with:

- **Subjective tasks:** Cultural context, lived experience
- **Novel domains:** Topics not in training data
- **Low-resource languages:** Limited training examples
- **Safety-critical:** Medical, legal, high-stakes
- **Evaluation benchmarks:** Cannot evaluate LLMs with LLM labels!
- **Expert knowledge:** Domain-specific expertise required

Key principle: Never use LLM annotations to evaluate LLM performance

Zero-Shot Annotation

Definition: Prompting LLM to annotate without examples

Example zero-shot prompt

```
Classify the sentiment of this text as POSITIVE, NEGATIVE, or NEUTRAL.
Text: ‘‘The movie was absolutely fantastic!’’
Sentiment:
```

Advantages:

- Simple to implement
- No example selection needed
- Fast iteration on prompts

Disadvantages:

- Relies on model's prior knowledge
- Less control over output format
- May not match your exact definitions

Few-Shot In-Context Learning

Definition: Provide examples in the prompt

Example few-shot prompt

```
Classify sentiment as POSITIVE, NEGATIVE, or NEUTRAL.
Text: ‘‘I love this!’’ → POSITIVE
Text: ‘‘Terrible experience.’’ → NEGATIVE
Text: ‘‘It was okay.’’ → NEUTRAL
Text: ‘‘Best purchase ever!’’ →
```

Benefits:

- Shows model exactly what you want
- Demonstrates output format
- Calibrates to your definitions

Choosing Examples for Few-Shot

Example selection matters

Best practices:

1. **Coverage:** Include examples for each category
2. **Clarity:** Use unambiguous examples
3. **Balance:** Similar number per category
4. **Relevance:** Examples similar to target data
5. **Edge cases:** Include difficult examples

How many examples?

- 2–5 examples per category typically works
- More isn't always better (context window limits)
- Quality over quantity

Prompt Design Patterns

Pattern 1: Role assignment

``You are an expert linguist annotating text for named entities...''

Pattern 2: Step-by-step reasoning

``First, identify all proper nouns. Then, classify each as PERSON, ORG, or LOC...''

Pattern 3: Format specification

``Return your answer as JSON: {"entities": [...]}''

Pattern 4: Constraint emphasis

``Only annotate entities explicitly mentioned. Do NOT infer entities.''

Prompt Best Practices

Guidelines for effective annotation prompts:

1. **Be explicit about output format** — Don't just ask for an answer; specify the exact format
2. **Include edge cases in few-shot examples** — If sarcasm is tricky, include a sarcasm example
3. **Use consistent formatting** — If examples use "$\rightarrow$ label", maintain that throughout
4. **Consider chain-of-thought** — Ask the model to reason before giving the final label
5. **Test on validation data** — Always validate before running at scale

Remember

Small changes to prompts can significantly affect annotation quality.

Structured Output Example

Request JSON output for reliable parsing:

```
Prompt:
Extract named entities from the text. Return as JSON.

Text: "Apple CEO Tim Cook announced the iPhone 16 in
Cupertino yesterday."

Output format:
{
  "entities": [
    {"text": "...", "start": N, "end": N,
     "label": "PER|ORG|LOC|DATE|PRODUCT"}
  ]
}
```

Model Selection for Annotation

Common choices:

- **GPT-4/GPT-4o:** Highest quality, most expensive
- **GPT-3.5:** Good balance of quality and cost
- **Claude 3.5 Sonnet:** Strong reasoning, good at instructions
- **Claude 3 Haiku:** Fast and cheap, good for simple tasks
- **Llama 3:** Open source, can run locally
- **Mistral:** Efficient open-source option

Selection factors:

- Task complexity
- Budget constraints
- Data privacy requirements
- Latency requirements

Cost Estimation

API pricing (approximate, as of 2025):

Model	Input ($/1M tokens)	Output ($/1M tokens)
GPT-4o	$2.50	$10.00
GPT-3.5	$0.50	$1.50
Claude 3.5 Sonnet	$3.00	$15.00
Claude 3 Haiku	$0.25	$1.25

Example calculation:

- 10,000 texts, avg 200 tokens each = 2M input tokens
- Output ~50 tokens each = 500K output tokens
- GPT-4o cost: $2 \times \$2.50 + 0.5 \times \$10 = \$10$

Cost Optimization Strategies

Reduce costs while maintaining quality:

1. **Model selection:** Use cheaper models for simple tasks
2. **Prompt efficiency:** Shorter prompts, fewer examples
3. **Batching:** Annotate multiple items per API call
4. **Caching:** Store results for repeated queries
5. **Filtering:** Pre-filter obvious cases
6. **Sampling:** LLM annotate subset, extrapolate

Batching example:
Instead of 100 API calls for 100 texts, send 10 calls with 10 texts each

Implementing LLM Annotation

Basic workflow:

1. Design prompt (zero-shot or few-shot)
2. Test on small sample
3. Validate against human annotations
4. Iterate on prompt if needed
5. Run on full dataset
6. Post-process and validate output

Key libraries:

- OpenAI Python SDK
- Anthropic Python SDK
- LangChain (abstraction layer)
- Hugging Face Transformers (local models)

Quality Assurance

Always validate LLM annotations:

Validation approaches:

1. **Sample review:** Human checks random subset
2. **Agreement:** Compare to gold standard annotations
3. **Consistency:** Run same prompt twice, compare
4. **Edge cases:** Test known difficult examples

Common issues:

- Output format violations
- Hallucinated annotations
- Position bias
- Inconsistent boundary decisions

Human-LLM Agreement

Measuring how well LLM matches humans:

Process:

1. Annotate subset with both humans and LLM
2. Calculate agreement metrics (Kappa, F1)
3. Analyze disagreements
4. Refine prompt based on errors

Typical targets:

- Simple classification: Kappa > 0.8 achievable
- Sequence labeling: Entity F1 > 0.7
- Complex tasks: May need human verification

Hybrid Approaches

Best of both worlds: LLM + Human

LLM pre-annotation + human correction:

1. LLM generates initial annotations
2. Humans review and correct
3. Faster than annotation from scratch
4. Maintains human quality control

Confidence-based routing:

1. LLM annotates with confidence score
2. High confidence: accept automatically
3. Low confidence: route to human

Key Takeaways

1. **LLM annotation** can be 10–100x cheaper than human annotation
2. **Zero-shot** is simple but less controlled; **few-shot** gives more precision
3. **Prompt design** significantly impacts quality
4. **Model selection** depends on task complexity and budget
5. **Always validate** LLM annotations against human judgments
6. **Hybrid approaches** combine LLM efficiency with human quality

Questions?

Office Hours: Wednesdays 1–3pm, Volen 109

jinzhao@brandeis.edu