COSI-230B: Natural Language Annotation for Machine Learning

Lecture 17: Instruction Annotation & Instruction-Tuning Datasets

Jin Zhao

Brandeis University

March 30, 2026

Today's Agenda

Part I: Foundations

- Instruction annotation as task contract
- Anatomy of an instruction example
- Instruction schema design best practices

Part II: Paper Deep Dives

- Natural Instructions (Mishra et al.)
- Super-NaturalInstructions (Wang et al.)
- T0 and zero-shot generalization (Sanh et al.)
- Flan Collection (Longpre et al.)
- Self-Instruct & Alpaca
- LIMA: Less Is More

Part III: Artifacts & Demos

- Template leakage: theory + concrete example
- Live demo: template shift breaks performance
- Live demo: paraphrase robustness test
- Instruction QA workflow
- Mitigations

Goal: Understand instruction data as task specification, not just prompt engineering.

Part I

Foundations of Instruction Annotation

Instruction Annotation = Specifying a Task Contract

The Conceptual Inversion

Classical annotation attaches labels *to data*.
Instruction annotation attaches *a task definition to many data*.

The hard part: Not writing *a* prompt—making an instruction distribution that yields **generalizable behavior**, not template overfitting.

> *"Instruction tuning does not magically confer generalization. It shifts the generalization problem: from unseen inputs to unseen tasks and formulations."*

Anatomy of an Instruction Example

Example: Sentiment Classification

Task definition: Given a movie review, classify the sentiment as positive or negative.

Input: "This film was a masterpiece of storytelling and visual artistry."

Constraints: Output only "positive" or "negative". Do not explain.

Output: positive

Instruction schema (standardized fields):

- **Task definition:** What the model should do
- **Input:** The specific instance
- **Constraints:** Output format, length, style requirements
- **Examples** (optional): Few-shot demonstrations
- **Output:** The expected response

Mishra et al. (2022). Natural Instructions. ACL. https://aclanthology.org/2022.acl-long.244/

Instruction Schema Design: Best Practices

Component	Natural Instr.	Super-Natural	Flan	Self-Instruct
Task definition	Verbose, multi-sentence	Standardized fields	Per-template prompt	Seed-generated
Pos/neg examples	Yes, both included	Yes, both included	No (implicit)	No
Constraints	Explicit field	Explicit field	Embedded in prompt	Implicit
Output format	Free text	Category-specific	Category-specific	Free text
Template count	1 per task	1 per task	∼10 per task	1 per task

Design principles:

1. **Separate intent from format:** The task definition should describe *what*, not *how to phrase it*
2. **Explicit constraints reduce ambiguity:** “Output only X or Y” prevents verbose answers
3. **Negative examples prevent over-generalization:** Show what *not* to do

Part II

Paper Deep Dives

Natural Instructions: Unified Schema for Many Tasks

Mishra et al. (2022): Benchmark generalization to *new tasks* given natural language instructions.

Key design decisions:

- Standardized instruction format
- Positive and negative examples
- Explicitly tests *cross-task* generalization
- Tasks from existing NLP datasets

What it revealed:

- Models can follow novel instructions
- Performance depends heavily on instruction wording
- Template/format choices leak into predictions
- Few-shot examples help but introduce their own biases

Mishra et al. (2022). Natural Instructions: Benchmarking Generalization to New Tasks from Natural Language Instructions. ACL. `https://aclanthology.org/2022.acl-long.244/`

Natural Instructions: Key Findings

Dataset: 61 tasks drawn from existing NLP benchmarks (QA, classification, generation, etc.)

Quantitative findings:

- GPT-3 achieves ∼55% RougeL on unseen tasks with instructions alone (vs. ∼22% without instructions)
- Adding 2 positive examples improves performance by ∼12 points on average
- Adding negative examples yields a further ∼4 point gain
- Performance variance across tasks is *very* high ($\sigma > 20$ points)

Annotation takeaways:

- Instruction quality matters more than instruction length
- Negative examples are under-used but highly informative
- Task decomposition (splitting complex tasks) consistently helps
- Human agreement on instruction clarity is moderate ($\kappa \approx 0.6$)

Lesson for annotators

Writing instructions is itself an annotation task that requires calibration, guidelines, and inter-annotator agreement measurement

Super-NaturalInstructions: Scaling Instruction Generalization

Wang et al. (2022): Scale to 1600+ NLP tasks with declarative instructions.

- Tests **generalization targets**: unseen tasks vs. unseen templates vs. unseen languages
- Shows that **instruction coverage** matters more than model size (up to a point)
- Formalizes "instruction generalization" as a benchmark

Generalization Target Taxonomy

Unseen tasks	New task types not in training
Unseen templates	Same task, different instruction wording
Unseen languages	Same task, different language

Annotation implication: How you split train/test determines what "generalization" means.

Wang et al. (2022). Super-NaturalInstructions. EMNLP. `https://aclanthology.org/2022.emnlp-main.340/`

Super-NaturalInstructions: Key Findings

Scale: 1,616 tasks, 76 task types, 55 languages, community-contributed via GitHub

Quantitative findings:

- Tk-Instruct (11B) trained on 757 tasks outperforms InstructGPT (175B) on held-out tasks
- Performance scales log-linearly with number of training tasks up to ∼800 tasks
- Cross-lingual transfer: 83% of English-only performance when tested on non-English tasks
- Diminishing returns beyond ∼64 examples per task

Community annotation process:

- Crowd-sourced via structured schema on GitHub
- Each task requires: definition, 2+ positive examples, 2+ negative examples, constraints
- Automated validation checks on submission
- Reveals tension: broad coverage vs. quality control at scale

Super-NaturalInstructions: Key Findings (cont.)

Key insight

Task diversity matters more than per-task data volume. 100 diverse tasks beat 10 tasks with $10\times$ data each.

T0 and the Zero-Shot Generalization Story

Sanh et al. (2022): Multitask prompted training enables zero-shot task generalization.

Design:

- Fine-tune T5-11B on a diverse mixture of prompted datasets (36 datasets, 62 tasks)
- Each task has multiple human-written prompt templates via PromptSource
- Evaluate zero-shot on 11 held-out datasets

Key results:

- T0 (11B) matches or exceeds GPT-3 (175B) on 9 of 11 held-out tasks
- Prompt template diversity is critical: more templates $\Rightarrow$ better generalization
- Median performance drops $\sim$15 points when using worst vs. best template per task

PromptSource annotation:

- Collaborative tool for writing and sharing prompt templates
- 2,073 prompts across 177 datasets
- Templates written by NLP researchers (not crowdworkers)
- Each template maps dataset fields to natural language

T0 and the Zero-Shot Generalization Story (cont.)

Template sensitivity

A single task can range from 30% to 80% accuracy depending on which template is used—template choice is a first-class annotation decision.

Sanh et al. (2022). Multitask Prompted Training Enables Zero-Shot Task Generalization. ICLR.

`https://arxiv.org/abs/2110.08207`

Flan: Design Decisions That Matter

Longpre et al. (2023): The Flan Collection—designing data and methods for effective instruction tuning.

Engineering decisions that affect outcomes:

1. **Task balancing:** Upsampling rare tasks, downsampling dominant ones
2. **Template diversity:** Multiple prompt templates per task
3. **Enrichment:** Adding chain-of-thought variants, few-shot examples
4. **Mixing ratios:** Proportion of zero-shot vs. few-shot vs. CoT

Key Evidence

Instruction tuning outcomes depend heavily on *mixing and enrichment decisions*—this is "annotation mixture engineering," not just training.

Longpre et al. (2023). The Flan Collection. arXiv:2301.13688. `https://arxiv.org/abs/2301.13688`

Flan: Ablation Results and Numbers

Flan 2022 merges Flan 2021, P3 (T0's data), Super-NaturalInstructions, and new dialog/CoT tasks.

Key ablation results:

- Mixing 1,836 tasks: +9.4% over Flan 2021 on held-out MMLU
- Adding CoT data: +2.1% on reasoning benchmarks (BBH)
- Template diversity (10 per task vs. 1): +4.2% on average
- Few-shot enrichment: +1.8% beyond zero-shot only
- Task balancing via examples-proportional mixing: +2.5% on rare task types

Mixture recipe:

- 50% zero-shot prompted
- 25% few-shot prompted
- 25% chain-of-thought
- Examples-proportional mixing with cap at 30K per task
- Input inversion: ∼10% of examples have input/output swapped for robustness

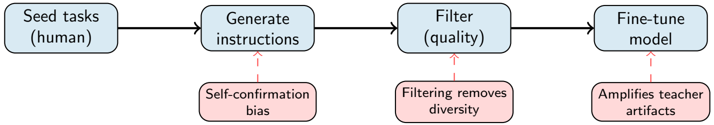
Flan: Ablation Results and Numbers (cont.)

Takeaway

Instruction tuning is as much about *data engineering* (mixing, balancing, enrichment) as it is about the model or the training algorithm.

Self-Instruct: Synthesize–Filter–Fine-Tune

Wang et al. (2023): Aligning Language Models with Self-Generated Instructions.



Where artifacts creep in:

- Prompting seeds shape the task distribution (seed bias)
- Filtering for quality/validity removes diversity
- Generated outputs inherit teacher model's style and errors

Wang et al. (2023). Self-Instruct. ACL. `https://aclanthology.org/2023.acl-long.754/`

Self-Instruct: Detailed Findings

Pipeline numbers:

- 175 human-written seed tasks $\rightarrow$ 82K generated instructions (after filtering)
- ROUGE-L filtering threshold: discard if > 0.7 overlap with existing instructions
- 52K instructions survive quality filtering ($\sim$63% pass rate)
- Human evaluation: 54% of generated instructions are valid and novel

Quality analysis:

- Valid instruction + correct output: 54%
- Valid instruction + incorrect output: 12%
- Invalid / nonsensical instruction: 34%
- Distribution heavily skewed toward classification and generation tasks

Impact on downstream model:

- GPT-3 + Self-Instruct tuning nearly matches InstructGPT on Super-NaturalInstructions
- Biggest gains on creative generation tasks
- Smallest gains on tasks requiring factual accuracy
- Seed task diversity directly predicts output diversity

Wang et al. (2023). Self-Instruct: Aligning Language Models with Self-Generated Instructions. ACL

Alpaca: “Semi-Synthetic Instruction Tuning”

Taori et al. (2023): Use GPT-3.5 to generate 52K instruction–output pairs from 175 seed tasks, then fine-tune LLaMA.

Pipeline:

1. Start with 175 human-written seed tasks
2. Prompt GPT-3.5 to generate new tasks + outputs
3. Light filtering for format/length
4. Fine-tune LLaMA-7B on result

Where annotation artifacts enter:

- Seed task distribution shapes output distribution
- GPT-3.5’s style becomes the “ground truth”
- No human verification of generated outputs
- Template artifacts from the generation prompt

Bridge to Module 5 (Synthetic Annotation): Alpaca shows synthetic data is partly annotation, partly generation.

Taori et al. (2023). Alpaca: A Strong, Replicable Instruction-Following Model. Stanford CRFM.

LIMA: Less Is More for Alignment

Zhou et al. (2023): LIMA—fine-tune LLaMA-65B on only **1,000 carefully curated examples**.

Key results:

- 1,000 examples ≈ comparable to 52K Alpaca examples in human preference evaluation
- LIMA preferred over Alpaca 73% of the time
- LIMA ties or beats GPT-4 (text-davinci-003) on 43% of prompts
- Adding 30 carefully written dialog examples enables multi-turn conversation

Data composition:

- 750 from Stack Exchange / wikiHow / Reddit (top answers)
- 250 manually written by the authors

The "Superficial Alignment Hypothesis"

A model's knowledge and capabilities are learned almost entirely during pretraining. Alignment tuning teaches the model the *format* and *style* of interaction, not new knowledge.

LIMA: Less Is More for Alignment (cont.)

Annotation implication:

- Quality $\gg$ quantity for instruction data
- Careful curation of 1K examples can rival 52K noisy ones
- Selection criteria matter more than generation volume

Zhou et al. (2023). LIMA: Less Is More for Alignment. NeurIPS. `https://arxiv.org/abs/2305.11206`

Part III

Artifacts, Demos, and Quality Assurance

Core Artifact: Template Leakage

Definition

Template leakage: When instruction tokens (template text) correlate with the label, a model can learn "what template implies" rather than "what the input means."

Example:

- Template A (for positive): "Say POSITIVE:" + review text
- Template B (for negative): "Say NEGATIVE:" + review text
- A classifier can achieve high accuracy by reading only the template!

Why decoder-only LMs are especially sensitive:

- Instruction text and output share the same token sequence
- Attention can freely attend to template tokens
- Spurious correlations in training are amplified at scale

Mitigation: Randomize templates, test with paraphrased instructions, audit template–label correlations.

Template Leakage: A Concrete Walkthrough

Scenario: NLI dataset with 3 labels. An annotator writes one template per label:

Templates used during instruction tuning

Entailment: "Given the premise, does the hypothesis follow? Answer yes."
Contradiction: "Given the premise, does the hypothesis conflict? Answer no."
Neutral: "Given the premise, is the hypothesis related? Answer maybe."

Token–label correlation analysis:

Token	PMI with label	TF-IDF rank	Problem
"follow"	1.0 (entailment)	#1	Perfectly predictive
"conflict"	1.0 (contradiction)	#1	Perfectly predictive
"related"	1.0 (neutral)	#1	Perfectly predictive
"yes"	1.0 (entailment)	#2	Label embedded in template
"no"	1.0 (contradiction)	#2	Label embedded in template

Template Leakage: A Concrete Walkthrough (cont.)

Result

A model trained on this data achieves >95% accuracy by attending to template tokens alone. On a *neutral* template ("Classify the relationship:") accuracy drops to ∼40%.

Why Decoder-Only LMs Are Sensitive to Formatting

Encoder-only (e.g., BERT):

- Input and output are separated
- Classification head ignores instruction format
- Less susceptible to template leakage

Decoder-only (e.g., GPT, LLaMA):

- Instruction + output in one sequence
- Autoregressive attention sees all prior tokens
- Template tokens directly influence generation

Practical Consequence

Format adherence vs. task quality becomes a real tradeoff. Instruction schema choices (input/output format, constraints, examples) directly shape modeling outcomes.

Live Demo 1: Template Shift Breaks Performance

```python
import random
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

random.seed(0)

def gen(n, template_style):
    X, y = [], []
    for _ in range(n):
        label = random.choice(["POS","NEG"])
        text = random.choice(["great␣movie","terrible␣movie",
                             "love␣it","hate␣it"])
        if template_style == "leaky":  # template correlates with label
            instr = "Say␣POSITIVE:" if label=="POS" else "Say␣NEGATIVE:"
        else:  # neutral template (no label-coded token)
            instr = "Classify␣sentiment:"
        X.append(instr + "␣" + text)
        y.append(label)
    return X, y

Xtr,ytr = gen(800,"leaky")
Xte1,yte1 = gen(300,"leaky")     # in-domain
Xte2,yte2 = gen(300,"clean")     # template-shifted

vec = TfidfVectorizer()
clf = LogisticRegression(max_iter=2000)
clf.fit(vec.fit_transform(Xtr), ytr)
print("in-domain:", accuracy_score(yte1, clf.predict(vec.transform(Xte1))))
print("template-shift:", accuracy_score(yte2, clf.predict(vec.transform(Xte2))))
```

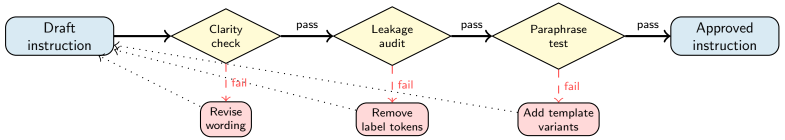
Live Demo 2: Paraphrase Robustness Test

```python
import random
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

random.seed(42)
reviews = ["great␣movie","terrible␣movie","love␣it","hate␣it",
           "amazing␣film","awful␣movie","enjoyed␣it","boring␣film"]
labels  = ["POS","NEG","POS","NEG","POS","NEG","POS","NEG"]

# Train with ONE template phrasing
templates_train = ["Classify␣the␣sentiment␣of␣this␣review:"]
# Test with PARAPHRASED templates (same meaning, different tokens)
templates_test  = ["What␣is␣the␣sentiment?",
                   "Determine␣whether␣this␣is␣positive␣or␣negative:",
                   "Read␣the␣review␣and␣output␣the␣sentiment:"]

def make_data(templates, n=600):
    X, y = [], []
    for _ in range(n):
        idx = random.randint(0, len(reviews)-1)
        t = random.choice(templates)
        X.append(t + "␣" + reviews[idx])
        y.append(labels[idx])
    return X, y
```

Live Demo 2: Paraphrase Robustness Test (cont.)

```
Xtr, ytr = make_data(templates_train, 600)
Xte_same, yte_same = make_data(templates_train, 200)
Xte_para, yte_para = make_data(templates_test, 200)

vec = TfidfVectorizer()
clf = LogisticRegression(max_iter=2000)
clf.fit(vec.fit_transform(Xtr), ytr)
print("same␣template:", accuracy_score(yte_same,
        clf.predict(vec.transform(Xte_same))))
print("paraphrased:␣␣", accuracy_score(yte_para,
        clf.predict(vec.transform(Xte_para))))
# Observe: even without leaky labels, template-specific tokens
# become spurious features when only one template is used.
```

Instruction QA Workflow



Stage details:

1. **Clarity check:** Can a non-expert understand the task from the instruction alone? (Human eval, 2 reviewers)
2. **Leakage audit:** Compute PMI between template tokens and labels. Flag if any token has PMI > 0.5 with a single label.
3. **Paraphrase test:** Rewrite the instruction 3 ways. If performance drops >10% on paraphrased versions, the instruction is over-fitted.
4. **Approval:** Instruction enters the final dataset with all template variants.

Mitigations for Template Leakage

1. **Template randomization:** Use multiple templates per task; ensure no template–label correlation
2. **Paraphrase tests:** Evaluate with rephrased instructions to test robustness
3. **Mixture audits:** Check task/template proportions; avoid systematic imbalances
4. **Cross-template evaluation:** Train on template A, test on template B
5. **Instruction QA workflow:**
    - Human reviewers check instructions for clarity and leakage
    - Edge-case checklists: adversarial inputs, boundary cases
    - Automated checks: token overlap between template and label

Longpre et al. (2023) show that Flan's template mixing improves robustness. `https://arxiv.org/abs/2301.13688`

Readings

Required readings for Lecture 18:

1. Wang et al. (2023). *Self-Instruct: Aligning Language Models with Self-Generated Instructions.* ACL 2023. **Sections 1–4.**
2. Longpre et al. (2023). *The Flan Collection: Designing Data and Methods for Effective Instruction Tuning.* arXiv:2301.13688. **Sections 1–3.**

Optional / recommended readings:

- Zhou et al. (2023). *LIMA: Less Is More for Alignment.* NeurIPS.
- Sanh et al. (2022). *Multitask Prompted Training Enables Zero-Shot Task Generalization.* ICLR.
- Bach et al. (2022). *PromptSource: An Integrated Development Environment for Natural Language Prompts.* ACL Demo.
- Mishra et al. (2022). *Natural Instructions.* ACL.