COSI-230B: Natural Language Annotation for Machine Learning

Lecture 9: Inter-Annotator Agreement II

Jin Zhao

Brandeis University

February 23, 2026

Today's Agenda

1. Review of Cohen's Kappa
2. Fleiss' Kappa (multiple annotators)
3. Crowdsourcing and WAWA

Review: Cohen's Kappa

For two annotators:

$$\kappa = \frac{A_o - A_e}{1 - A_e}$$

Limitations:

- Only works for exactly 2 annotators
- Assumes same 2 annotators for all items
- Categorical labels only

Today: Extensions for more complex scenarios

Cohen's Kappa: Quick Example

Binary labels, 30 items

	Maya: Yes	Maya: No	Total
Jin: Yes	12	3	15
Jin: No	2	13	15
Total	14	16	30

Goal: Compute A_o, A_e, then κ

Cohen's Kappa: Step-by-Step

	Maya: Yes	Maya: No	Total
Jin: Yes	12	3	15
Jin: No	2	13	15
Total	14	16	30

Observed agreement:

$$A_o = \frac{12 + 13}{30} = \frac{25}{30} = 0.833$$

Expected agreement:

- P(both Yes) $= \frac{15}{30} \times \frac{14}{30} = 0.50 \times 0.467 = 0.233$
- P(both No) $= \frac{15}{30} \times \frac{16}{30} = 0.50 \times 0.533 = 0.267$

$$A_e = 0.233 + 0.267 = 0.50$$

Kappa:

$$\kappa = \frac{0.833 - 0.50}{1 - 0.50} = 0.666$$

Why Fleiss' Kappa?

Motivation: Two-annotator setups are common, but they are not the only case

Common situations:

- 3+ annotators label the same items
- Annotators rotate across batches (different pairs per item)
- You want one agreement number across the whole team

What it is:

- Chance-corrected agreement for multiple annotators
- Generalizes Cohen's Kappa beyond 2 raters
- Works with categorical labels

Why Rotation Needs Fleiss' Kappa

Cohen's Kappa assumes the same two annotators for all items

When annotators rotate:

- Item 1 might be labeled by (Jin, Maya)
- Item 2 might be labeled by (Jin, Sam)
- Item 3 might be labeled by (Maya, Alex)

Problem: There is no single, fixed annotator pair

Why Cohen's fails:

- It needs one confusion matrix for the same two raters
- Mixed pairs break the chance-agreement calculation

Fleiss' Kappa: aggregates agreement across a pool of annotators

Fleiss' Kappa fixes this by treating the pool as a set of interchangeable annotators and computing agreement per item based on how many annotators chose each label, then aggregating across items.

Fleiss' Kappa: Multiple Annotators

When you have 3+ annotators

Same formula structure:

$$\kappa = \frac{\bar{P} - \bar{P}_e}{1 - \bar{P}_e}$$

Key differences from Cohen's:

- Works with any number of annotators
- Different annotators can label different items
- More complex calculation of expected agreement

Assumption: Fixed number of annotators per item (e.g., always 3)

Fleiss' Kappa Calculation

For n items, k categories, r annotators per item:

Let n_{ij} = number of annotators who assigned item i to category j

Per-item agreement:

$$P_i = \frac{1}{r(r-1)} \sum_{j=1}^{k} n_{ij}(n_{ij} - 1)$$

Mean observed agreement:

$$\bar{P} = \frac{1}{n} \sum_{i=1}^{n} P_i$$

Expected agreement:

$$\bar{P}_e = \sum_{j=1}^{k} p_j^2$$

where p_j = proportion of all ratings in category j

Slide 9 in Plain Language

Core question: If several people label the same items, how much do they really agree beyond coincidence?

Setup:

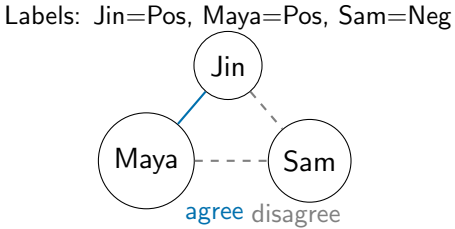
- n items (e.g., 100 articles)
- r annotators per item (e.g., 3 people)
- k categories (e.g., Positive/Neutral/Negative)

Per-Item Agreement (P_i)

For one item: Count how many annotators picked each category (n_{ij}).

Idea: Look at all pairs of annotators on that item.

- If two annotators chose the same label, that pair agrees.
- P_i = fraction of agreeing pairs out of all possible pairs.



Interpretation:

- $P_i = 1$ means everyone agreed.
- $P_i = 0$ means no two annotators agreed.

Observed Agreement ($\bar{P}$)

Simple average: Compute P_i for each item, then average across all items.

Meaning:

- $\bar{P}$ is the real, observed agreement rate.
- It tells you how often annotators agree in practice.

Expected Agreement ($\bar{P}_e$)

Chance agreement: If annotators guessed labels but kept the same overall label proportions, how often would they match by accident?

Let p_j be the overall proportion of label j.

- Example: 70% Positive, 20% Neutral, 10% Negative

Chance match for a label:

- Both pick Positive by chance: $0.7 \times 0.7 = 0.49$

Why We Need Chance Correction

Problem: High agreement can happen even without real consensus.

If one label dominates (e.g., 90% Positive), then chance agreement is already high.

Example:

- Observed agreement $= 80\%$
- Chance agreement $= 75\%$
- Real signal is small

Fleiss' Kappa: The Adjustment

Fleiss' Kappa compares:

$$\frac{\text{Observed} - \text{Chance}}{1 - \text{Chance}}$$

Meaning: How much better are we than random guessing?

Big Picture Summary

- P_i = agreement for one item
- $\bar{P}$ = average real agreement
- $\bar{P}_e$ = agreement expected by accident
- Kappa adjusts for label imbalance

Understanding P_i: The Core Idea

Question: For one item, what fraction of annotator pairs agree?

$$P_i = \frac{\text{ordered agreeing pairs}}{\text{ordered total pairs}}$$

What is an ordered pair?

- Pick one annotator first, pick another second
- (Jin, Maya) and (Maya, Jin) count as **two** ordered pairs
- This is just a counting convention — the math works out the same

Why not unordered?

- You *can* use unordered pairs — you'd get $\binom{r}{2}$ in the denominator and $\binom{n_{ij}}{2}$ in the numerator
- Ordered pairs give the same ratio but simpler algebra: $r(r-1)$ and $n_{ij}(n_{ij}-1)$

Step 1: Counting Total Ordered Pairs

With r annotators, how many ordered pairs are there?

- First pick: r choices (any annotator)
- Second pick: $r - 1$ choices (any *other* annotator)
- Total ordered pairs $= r \times (r - 1)$

Example with $r = 3$ annotators (Jin, Maya, Sam):

(J,M) (J,S) (M,J) (M,S) (S,J) (S,M)

$$r(r-1) = 3 \times 2 = 6 \text{ ordered pairs}$$

This is the **denominator** of P_i.

Step 2: Counting Ordered Agreeing Pairs

An ordered pair agrees if both annotators chose the same label.

For category j: suppose n_{ij} annotators chose it.

- First pick from those n_{ij}: n_{ij} choices
- Second pick from the remaining: $n_{ij} - 1$ choices
- Ordered agreeing pairs for category $j = n_{ij}(n_{ij} - 1)$

Sum across all categories:

$$\text{Total agreeing pairs} = \sum_{j=1}^{k} n_{ij}(n_{ij} - 1)$$

This is the **numerator** of P_i.

Putting It Together: The P_i Formula

$$P_i = \frac{\sum_{j=1}^{k} n_{ij}(n_{ij}-1)}{r(r-1)} = \frac{1}{r(r-1)} \sum_{j=1}^{k} n_{ij}(n_{ij}-1)$$

Concrete example: 3 annotators label one item with categories Pos/Neg/Neu.
Suppose the labels are: Pos, Pos, Neg $\Rightarrow$ $n_{\text{Pos}} = 2,\ n_{\text{Neg}} = 1,\ n_{\text{Neu}} = 0$

Numerator (agreeing pairs):

- Cat. Pos: $2 \times 1 = 2$
- Cat. Neg: $1 \times 0 = 0$
- Cat. Neu: $0 \times 0 = 0$
- Total $= 2$

Denominator (total pairs):

- $r(r-1) = 3 \times 2 = 6$

Result:

$$P_i = \frac{2}{6} = 0.33$$

Check: 3 pairs total — only the (Pos,Pos) pair agrees $\Rightarrow \frac{1}{3} = 0.33$ ✓

Fleiss' Kappa: Worked Example (Data)

5 items, 3 categories (Pos/Neg/Neu), 3 annotators each

Item	Pos	Neg	Neu	P_i
1	3	0	0	1.00
2	2	1	0	0.33
3	0	3	0	1.00
4	1	1	1	0.00
5	0	2	1	0.33

Compute each P_i using $P_i = \frac{1}{r(r-1)} \sum_j n_{ij}(n_{ij}-1)$, $r = 3$, so $r(r-1) = 6$:

- $P_1 = \frac{3 \cdot 2 + 0 + 0}{6} = 1.00$ $\quad P_2 = \frac{2 \cdot 1 + 1 \cdot 0 + 0}{6} = 0.33$ $\quad P_3 = \frac{0 + 3 \cdot 2 + 0}{6} = 1.00$
- $P_4 = \frac{1 \cdot 0 + 1 \cdot 0 + 1 \cdot 0}{6} = 0.00$ $\quad P_5 = \frac{0 + 2 \cdot 1 + 1 \cdot 0}{6} = 0.33$

Mean observed agreement: $\bar{P} = \frac{1}{5}(1 + 0.33 + 1 + 0 + 0.33) = 0.533$

Fleiss' Kappa: Step 3a (Expected Agreement)

Item	Pos	Neg	Neu	P_i
1	3	0	0	1.00
2	2	1	0	0.33
3	0	3	0	1.00
4	1	1	1	0.00
5	0	2	1	0.33

Category proportions: $p_j = \frac{\text{total in cat. } j}{n \times r}$, total $= 15$

- $p_{\text{Pos}} = \frac{3+2+0+1+0}{15} = \frac{6}{15} = 0.40$
- $p_{\text{Neg}} = \frac{0+1+3+1+2}{15} = \frac{7}{15} = 0.467$
- $p_{\text{Neu}} = \frac{0+0+0+1+1}{15} = \frac{2}{15} = 0.133$

Formula: $\bar{P}_e = \sum_{j=1}^{k} p_j^2$ (sum of squared proportions across all k categories)

Computation:

$$\bar{P}_e = p_{\text{Pos}}^2 + p_{\text{Neg}}^2 + p_{\text{Neu}}^2 = 0.40^2 + 0.467^2 + 0.133^2 = 0.16 + 0.218 + 0.018 = 0.396$$

Fleiss' Kappa: Step 3b (Kappa)

Formula:

$$\kappa = \frac{\bar{P} - \bar{P}_e}{1 - \bar{P}_e}$$

$\bar{P}$ (observed agreement from Step 2) $= 0.533$

$\bar{P}_e$ (expected agreement from Step 3a) $= 0.396$

Computation:

$$\kappa = \frac{0.533 - 0.396}{1 - 0.396} = \frac{0.137}{0.604} = 0.228$$

Interpretation: Fair agreement (only modestly above chance)

Use Python: `statsmodels.stats.inter_rater.fleiss_kappa`

Interpreting Fleiss' κ: Landis & Koch Guidelines

Fleiss' κ	Interpretation
< 0	Poor (worse than chance)
$0.00 - 0.20$	Slight agreement
$0.21 - 0.40$	Fair agreement
$0.41 - 0.60$	Moderate agreement
$0.61 - 0.80$	Substantial agreement
$0.81 - 1.00$	Almost perfect agreement

Rules of thumb for NLP annotation:

- $\kappa \geq 0.80$: Reliable — safe to train models on this data
- $0.67 \leq \kappa < 0.80$: Acceptable for many tasks — review guidelines
- $\kappa < 0.67$: Problematic — revise guidelines, retrain annotators, or simplify the task

Our worked example: $\kappa = 0.228$ (fair) — guidelines likely need work

Caveats When Interpreting κ

κ values don't tell the whole story:

1. **Prevalence problem:** When one category dominates, κ can be low even with high observed agreement (we'll see this later with WAWA)
2. **Number of categories matters:** More categories $\rightarrow$ lower expected agreement $\rightarrow$ higher κ for the same observed agreement
3. **Task difficulty varies:** Sentiment ($\kappa \approx 0.7$) is harder to agree on than POS tagging ($\kappa \approx 0.95$)
4. **Always report alongside:**
   - Number of items, annotators, and categories
   - Per-category agreement (some labels may be harder than others)
   - The raw observed agreement $\bar{P}$

Your Turn: Fleiss' Kappa

4 items, 3 categories (Pos/Neg/Neu), 3 annotators each

Item	Pos	Neg	Neu
1	3	0	0
2	1	2	0
3	0	3	0
4	1	1	1

Compute: each P_i, then $\bar{P}$, $\bar{P}_e$, and κ

Formulas (Hint)

$P_i = \frac{1}{r(r-1)} \sum_j n_{ij}(n_{ij} - 1)$

$\bar{P} = \frac{1}{n} \sum_i P_i$

$\bar{P}_e = \sum_j p_j^2$

$\kappa = \frac{\bar{P} - \bar{P}_e}{1 - \bar{P}_e}$

Solution: Step 1 — Per-Item Agreement (P_i)

Item	Pos	Neg	Neu
1	3	0	0
2	1	2	0
3	0	3	0
4	1	1	1

$r = 3, \quad r(r-1) = 6$

$$P_i = \frac{1}{r(r-1)} \sum_j n_{ij}(n_{ij} - 1)$$

- $P_1 = \frac{3 \cdot 2 + 0 \cdot 0 + 0 \cdot 0}{6} = \frac{6}{6} = 1.00$
- $P_2 = \frac{1 \cdot 0 + 2 \cdot 1 + 0 \cdot 0}{6} = \frac{2}{6} = 0.33$
- $P_3 = \frac{0 \cdot 0 + 3 \cdot 2 + 0 \cdot 0}{6} = \frac{6}{6} = 1.00$
- $P_4 = \frac{1 \cdot 0 + 1 \cdot 0 + 1 \cdot 0}{6} = \frac{0}{6} = 0.00$

Mean observed agreement:

$$\bar{P} = \frac{1}{4}(1.00 + 0.33 + 1.00 + 0.00) = \frac{2.33}{4} = 0.583$$

Solution: Step 2 — Expected Agreement ($\bar{P}_e$)

Item	Pos	Neg	Neu
1	3	0	0
2	1	2	0
3	0	3	0
4	1	1	1

Total $= n \times r = 4 \times 3 = 12$

Formula: $\bar{P}_e = \sum_{j=1}^{k} p_j^2$

Computation:

Category proportions: $p_j = \frac{\text{total in cat. } j}{n \times r}$

- $p_{\text{Pos}} = \frac{3+1+0+1}{12} = \frac{5}{12} = 0.417$
- $p_{\text{Neg}} = \frac{0+2+3+1}{12} = \frac{6}{12} = 0.500$
- $p_{\text{Neu}} = \frac{0+0+0+1}{12} = \frac{1}{12} = 0.083$

$$\bar{P}_e = 0.417^2 + 0.500^2 + 0.083^2 = 0.174 + 0.250 + 0.007 = 0.431$$

Solution: Step 3 — Kappa (κ)

Formula:

$$\kappa = \frac{\bar{P} - \bar{P}_e}{1 - \bar{P}_e}$$

$$\bar{P} \text{ (observed agreement from Step 1)} = 0.583$$
$$\bar{P}_e \text{ (expected agreement from Step 2)} = 0.431$$

Computation:

$$\kappa = \frac{0.583 - 0.431}{1 - 0.431} = \frac{0.152}{0.569} = 0.267$$

Interpretation: Fair agreement — annotators agree modestly above chance

Why Crowdsourcing?

Traditional annotation: hire a small team of trained experts

- High quality per annotator
- But **slow** and **expensive**
- Bottleneck for large-scale NLP datasets

Crowdsourcing: distribute micro-tasks to a large pool of online workers

- Fast: thousands of labels in hours
- Cheap: pennies per label
- Scalable: hundreds of workers available on demand

The trade-off:

- Workers are untrained — variable quality
- Need **redundancy** (multiple labels per item) + **quality control**

Crowdsourcing in NLP: A Brief History

Crowdsourcing transformed how we build NLP datasets:

- **2008–2012:** Early adoption — Snow et al. (2008) showed crowd workers can match experts on many NLP tasks when aggregated
- **2012–2018:** Standard practice — SQuAD, SNLI, and most large benchmarks built via crowdsourcing
- **2018–present:** Still widely used, but increasingly combined with LLM-based annotation

Key insight from Snow et al.:

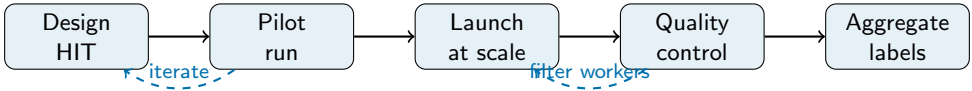
With enough redundancy (3–5 workers per item) and simple aggregation (majority vote), non-expert crowd annotations can rival expert quality.

Crowdsourcing Platforms

Platform	Strengths	Common NLP uses
Amazon MTurk	Large worker pool, flexible task design	Sentiment, NLI, QA, NER
Prolific	Academic-friendly, better demographics	Surveys, subjective tasks
Appen / Scale AI	Managed workforce, enterprise quality	Production ML pipelines
Label Studio / Doccano	Self-hosted, free	In-house annotation teams

In this course: We focus on the MTurk model (most studied in NLP research), but the principles apply to any platform.

The Crowdsourcing Annotation Pipeline



1. **Design HIT:** Clear instructions, examples, and UI
2. **Pilot run:** Small batch to test guidelines and catch problems
3. **Launch at scale:** Open to full worker pool
4. **Quality control:** Monitor agreement, filter bad workers
5. **Aggregate:** Majority vote or more sophisticated methods

Quality Challenges in Crowdsourcing

Not all workers are equal:

Common problems:

- **Spammers** — random clicks to collect payment
- **Lazy workers** — always pick the first option
- **Misunderstanding** — didn't read instructions
- **Genuine disagreement** — task is ambiguous

Common defenses:

- Gold / trap questions (known answers)
- Qualification tests before starting
- Redundancy (3–5 labels per item)
- Agreement-based worker filtering
- Time-on-task thresholds

The core question for this lecture

How do we **measure** worker quality when the annotation matrix is sparse and workers vary across items?

What Is Amazon Mechanical Turk (MTurk)?

Amazon Mechanical Turk is an online crowdsourcing marketplace where *requesters* post small tasks and *workers* complete them for pay.

How it works:

- Requesters create **HITs** (Human Intelligence Tasks)
- Each HIT is a micro-task: label an image, classify a sentence, rate a translation...
- Workers choose which HITs to do
- Payment is typically \$0.01–\$0.50 per HIT

Why NLP researchers use it:

- Thousands of annotations in hours, not weeks
- Access to a large, diverse worker pool
- Low cost compared to hiring expert annotators

The catch:

- Variable worker quality
- No guarantee the same workers label every item
- Need robust quality control

Why Not Fleiss' κ on MTurk?

Recall: Fleiss' κ allows *different* annotators per item (unlike Cohen's), but still assumes:

- Every item is labeled by exactly r annotators (fixed count)
- No missing data — every item gets all r labels
- The category matrix has no gaps

Crowdsourcing (e.g., MTurk) violates even these assumptions:

- Item 1 gets 5 labels, Item 2 gets 2 — r varies per item
- Workers drop out mid-task, creating missing data
- Some workers label 5 items, others label 500
- Overlap between any two workers may be tiny or zero

Problem: You cannot build the clean $n \times k$ category matrix that Fleiss' κ requires when r is not constant.

The Messy Reality of Crowdsourcing

Lab setting (Fleiss' κ works)

	Ann	Bob	Cat
Item 1	A	A	B
Item 2	B	B	B
Item 3	A	B	A
Item 4	A	A	A

- ✓ Fixed 3 annotators
- ✓ Every cell filled

MTurk setting (Fleiss' κ breaks)

	W1	W2	W3	W4	W5
Item 1	A	–	B	–	A
Item 2	–	B	–	B	–
Item 3	–	A	B	–	–
Item 4	A	–	–	A	–

- ✗ Workers vary per item
- ✗ Sparse, unbalanced matrix

Key insight: With hundreds of workers and thousands of items, most cells are empty.

κ and Class Imbalance

Another problem: Fleiss' κ can be *misleadingly low* when one class dominates.

Example: Binary task (Spam / Not Spam), 80% of items are Not Spam.

	Observed agreement	Fleiss' κ
Three workers agree 90% of the time	0.90	0.38

Why? P_e is already high (≈ 0.68) due to skew, so the gap $\bar{P} - \bar{P}_e$ shrinks.

The paradox

Workers are doing a good job, but κ says "fair agreement." In crowdsourcing, many real tasks are heavily skewed.

What Is WAWA?

WAWA = **W**orker-**A**greement **W**ith **A**ggregate

A simple, robust quality metric for crowdsourcing:

1. **Build the aggregate label** for each item
   (e.g., majority vote across all workers who labeled it)
2. **Compute each worker's agreement** with the aggregate
   $\text{WAWA}_w = \frac{\#\text{ items where worker } w \text{ matches aggregate}}{\#\text{ items worker } w \text{ labeled}}$
3. **Average** across all workers
   $\text{WAWA} = \frac{1}{W}\sum_{w=1}^{W} \text{WAWA}_w$

Properties:

- No need for fixed annotator sets — each worker is evaluated independently
- Handles sparse, unbalanced annotation matrices naturally
- Range: 0 (no agreement) to 1 (perfect agreement with consensus)

WAWA: Worked Example

Setup: 4 items, 3 workers (sparse assignments)

	W1	W2	W3	**Aggregate**
Item 1	Pos	Pos	–	Pos
Item 2	Neg	–	Neg	Neg
Item 3	–	Pos	Neg	? (tie $\rightarrow$ Pos)
Item 4	Pos	Neg	Pos	Pos

Step 1: Aggregate = majority vote per item

Step 2: Per-worker agreement with aggregate:

- W1 labeled items $\{1, 2, 4\}$: matches on $\{1, 4\} \Rightarrow \text{WAWA}_{W1} = \frac{2}{3} = 0.67$
- W2 labeled items $\{1, 3, 4\}$: matches on $\{1, 3\} \Rightarrow \text{WAWA}_{W2} = \frac{2}{3} = 0.67$
- W3 labeled items $\{2, 3, 4\}$: matches on $\{2, 4\} \Rightarrow \text{WAWA}_{W3} = \frac{2}{3} = 0.67$

Step 3: $\text{WAWA} = \frac{1}{3}(0.67 + 0.67 + 0.67) = \mathbf{0.67}$

Your Turn: WAWA

5 items, 4 workers (sparse assignments), labels Pos/Neg

	W1	W2	W3	W4
Item 1	Pos	Pos	–	Neg
Item 2	–	Neg	Neg	Neg
Item 3	Neg	–	Pos	Pos
Item 4	Neg	Pos	–	Neg
Item 5	Pos	Neg	Pos	–

Compute:

1. The **aggregate label** (majority vote) for each item
2. Each worker's **WAWA** score
3. The **overall WAWA**

Reminder

$$\text{WAWA}_w = \frac{\#\text{ items where worker } w \text{ matches aggregate}}{\#\text{ items worker } w \text{ labeled}} \qquad \text{WAWA} = \frac{1}{W}\sum_w \text{WAWA}_w$$

Solution: Step 1 — Aggregate Labels

Majority vote for each item (count Pos vs. Neg among available labels):

	W1	W2	W3	W4	Pos	Neg	**Aggregate**
Item 1	Pos	Pos	–	Neg	2	1	**Pos**
Item 2	–	Neg	Neg	Neg	0	3	**Neg**
Item 3	Neg	–	Pos	Pos	2	1	**Pos**
Item 4	Neg	Pos	–	Neg	1	2	**Neg**
Item 5	Pos	Neg	Pos	–	2	1	**Pos**

No ties in this example — majority vote is unambiguous for all items.

Solution: Step 2 — Per-Worker WAWA

Compare each worker's labels to the aggregate:

	W1		W2		W3		W4	
	Label	Match?	Label	Match?	Label	Match?	Label	Match?
Item 1 (Pos)	Pos	✓	Pos	✓	–	–	Neg	×
Item 2 (Neg)	–	–	Neg	✓	Neg	✓	Neg	✓
Item 3 (Pos)	Neg	×	–	–	Pos	✓	Pos	✓
Item 4 (Neg)	Neg	✓	Pos	×	–	–	Neg	✓
Item 5 (Pos)	Pos	✓	Neg	×	Pos	✓	–	–
Items labeled	4		4		3		4	
Matches	3		2		3		3	
WAWA$_w$	**0.75**		**0.50**		**1.00**		**0.75**	

Solution: Step 3 — Overall WAWA

Per-worker scores:

	W1	W2	W3	W4
WAWA_w	0.75	0.50	1.00	0.75

Overall WAWA:

$$\text{WAWA} = \frac{1}{4}(0.75 + 0.50 + 1.00 + 0.75) = \frac{3.00}{4} = \mathbf{0.75}$$

Interpretation:

- Overall WAWA $= 0.75$ — moderate agreement with consensus
- **W3** (1.00) is the most reliable worker
- **W2** (0.50) is concerning — disagreed with the majority on half the items
- **Action:** Review W2's work; consider removing and re-collecting those labels

Interpreting WAWA Scores

WAWA ranges from 0 to 1 (fraction of agreement with consensus):

Overall WAWA	**Interpretation**
≥ 0.90	Excellent — workers are highly consistent with consensus
$0.80 - 0.89$	Good — minor disagreements, acceptable for most tasks
$0.70 - 0.79$	Moderate — review guidelines or increase redundancy
$0.60 - 0.69$	Marginal — noticeable noise, consider filtering workers
< 0.60	Poor — significant quality problems

Per-worker WAWA is even more useful:

- Set a threshold (e.g., $\text{WAWA}_w \geq 0.70$) to keep reliable workers
- Workers below the threshold: remove their labels and re-collect
- Monitor trends over time — a dropping score may signal fatigue or confusion

Our worked example: WAWA $= 0.67$ (marginal) — all three workers show the same moderate agreement, suggesting the task itself may be ambiguous

WAWA: Practical Tips

(1) Thresholds depend on your task:

- Easy binary task (spam detection): expect WAWA ≥ 0.85
- Subjective task (sentiment, toxicity): WAWA ≥ 0.70 may be realistic
- If *all* workers score low, the problem is the task, not the workers

(2) WAWA is only as good as the aggregate:

- With very few labels per item, majority vote can be noisy
- More redundancy (5+ workers) $\rightarrow$ more stable aggregate $\rightarrow$ more meaningful WAWA

(3) Use WAWA alongside other checks:

- Gold / trap questions catch random clickers
- WAWA catches *systematically* wrong workers
- Time-on-task catches workers who rush without reading

WAWA vs. Fleiss' κ: Different Questions

	Fleiss' κ	**WAWA**
Question asked	Do annotators agree beyond chance?	Does each worker match the consensus?
Chance correction	Yes	No (not needed — aggregate serves as reference)
Annotator design	Fixed, fully crossed	Any overlap pattern
Missing data	Not allowed	Handled naturally
Class imbalance	Sensitive (can deflate)	Robust
Per-worker scores	No	Yes — can identify weak workers

Key distinction: κ measures *inter-rater reliability*; WAWA measures *individual worker quality* against a consensus standard.

When to Use Which

Use Fleiss' κ when...

- Small, trained annotator team
- Every annotator labels every item (or nearly so)
- You want to assess *task difficulty* or *guideline clarity*
- Lab or controlled setting

Use WAWA when...

- Many crowd workers with varying overlap
- Sparse annotation matrix
- You need to flag or filter low-quality workers
- You want a quick, interpretable quality check

Practical tip

In large-scale crowdsourcing pipelines, compute *both*:

- WAWA per worker $\rightarrow$ filter out unreliable workers
- κ on a small expert subset $\rightarrow$ validate guideline quality

Key Takeaways

1. **Fleiss' κ** extends Cohen's κ to multiple annotators
   - Allows different annotators per item, but requires fixed r per item
2. **Crowdsourcing** (e.g., MTurk) breaks the fixed-r assumption
   - Sparse, unbalanced annotation matrices
3. **WAWA** measures each worker's agreement with the consensus
   - Handles sparse data; gives per-worker quality scores
4. **Choose your metric based on your design:**
   - Lab setting with fixed team $\rightarrow$ Fleiss' κ
   - Crowdsourcing with variable workers $\rightarrow$ WAWA

Next time: Krippendorff's Alpha, sequence labeling agreement, and more

Questions?

Office Hours: Wednesdays 1-3pm, Volen 109

jinzhao@brandeis.edu