

Trustworthy Confidence

Knowing When LLMs Should Not Be Trusted

Advanced LLMs – Responsible LLMs

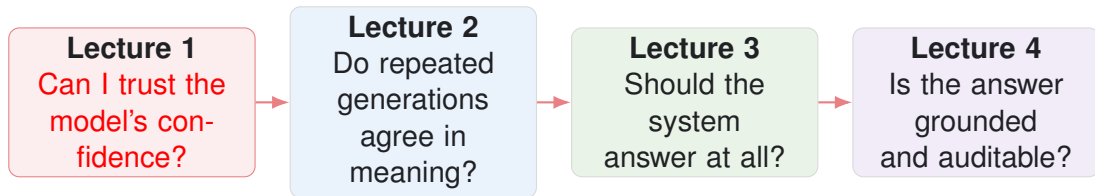


Sourish Dasgupta

Associate Professor, DAU, Gandhinagar

Lecture 1 of the Responsible Uncertainty block

Four-lecture Uncertainty Story



Question: when a model gives us a confidence signal, *what does that signal practically mean?*



Part I. Verbalized Confidence



Student

“I was ill and missed an assignment deadline by five days. I have a medical certificate. Am I automatically entitled to an extension?”

Hypothetical LLM assistant

“Yes. Students with a valid medical certificate are automatically entitled to a seven-day extension.”

It sounds confident.

Fluent
Specific
Authoritative

No numerical confidence score.

Only an answer that *looks* confident to a reader.



Why does the answer *feel* confident?

Fluency

No visible hesitation

Specificity

“seven-day
extension”

Authority cues

direct declar-
ative tone

A human reader can infer confidence from **style**.

A style cue is not a probability of correctness.

First distinction: perceived confidence $\neq$ measured confidence.



Numerical confidence score - but how?

No single universal “LLM confidence” score – need to **construct or elicit** one.

White-box score

logits, token probabilities, entropy

Verbalized confidence

ask the model “*How sure are you?*”

External score

a learned head, judge, or verifier

Takeway: Who produced the score, from what information, and by what procedure?



Let's *elicit* the score explicitly

We send a **second prompt** to the same hypothetical assistant.

Follow-up prompt

“How confident are you that your previous answer is correct? Give a number between 0% and 100%.”

LLM assistant

“I am 95% confident that the answer is correct.”

95%
self-reported



The 95% is **NOT** automatically any of these things

Is it
the probability of
the last token?

Is it
the product of
token probabilities?

Is it
a calibrated
probability of
correctness?

The model was asked to emit a number, and it emitted one. That is all we know so far.

A number in $[0, 1]$ is not automatically a trustworthy probability.



What would we *like* 95% to mean?

Possibility 1

“The **words** I generated were very probable.”

Possibility 2

“The answer feels very plausible to me.”

What we want

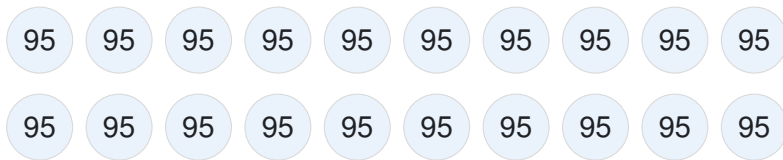
“Among answers like this, I am correct about 95% of the time.”

For Responsible AI, the 3rd interpretation is useful. Why?



How could we test whether “95%” deserves the 3rd interpretation?

Imagine we collect many previous answers for which the model also reported **95% confidence**.



For each answer, an **independent reference** tells us whether it was **correct** or **wrong**.

A confidence score becomes meaningful only when we compare it against what **Actually happened.**



Example: 20 answers, each reported as 95% confident

Suppose independent checking gives us:



Correct answers = $8 + 7 = 15$ out of 20.

$$\text{empirical accuracy} = \frac{15}{20} = 0.75.$$

The model said 95%; reality gave us 75%. That is **overconfidence**.



Calibration is needed!

For answers receiving confidence score p , we would like their long-run empirical correctness rate to be *about* p .



Formally:

$$\Pr(Y = 1 \mid S = p) = p.$$

Here $Y = 1$ means “the answer is correct” and S is the confidence score.



What calibration does **NOT** say

Calibration is a frequency statement over **groups** of predictions.

If 100 answers are all assigned 80% confidence and about 80 are correct, **that group** is well calibrated.

But for one particular answer with score 0.80, calibration does **not** tell us:

- ▶ *which specific* 20 answers are the *wrong* ones;
- ▶ whether the current answer **used the right evidence**;
- ▶ whether a **citation** actually **supports** the claim;
- ▶ whether the question is high-stakes enough to require verification anyway.

Calibration $\neq$ a certificate of **instance-level truth or grounding.**



Part II. White-box Confidence



What's under the Auto-regressive hood?

At generation step t , the model has the prompt x and previously generated tokens $y_{<t}$.



Formally:

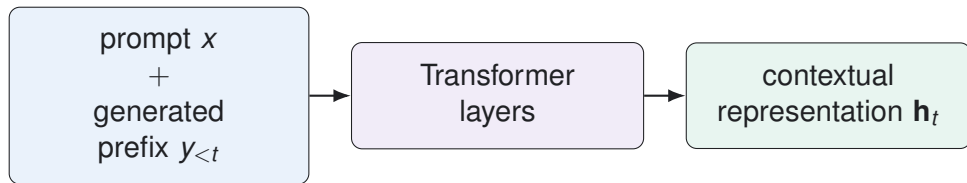
$$p_{\theta}(y_t \mid x, y_{<t}).$$

The model's native probability: *which token comes next?*



Let us open the “LLM” box

The model receives x (original prompt) + $y_{<t}$ everything generated so far.



Hidden contextual representation (d -dimensional embedding):

$$\mathbf{h}_t = \text{Transformer}_{\theta}(x, y_{<t}); \mathbf{h} \in \mathbb{R}^d$$



The LM **head** converts context into vocabulary scores

We need a linear map from

$$\mathbb{R}^d \longrightarrow \mathbb{R}^{|\mathcal{V}|}.$$

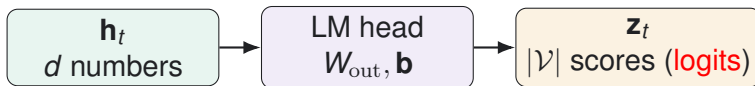
The language-model output **head** provides this map.

$$\mathbf{z}_t = \mathbf{W}_{\text{out}} \mathbf{h}_t + \mathbf{b}$$

where

$$\mathbf{W}_{\text{out}} \in \mathbb{R}^{|\mathcal{V}| \times d}, \quad \mathbf{b} \in \mathbb{R}^{|\mathcal{V}|}.$$

Therefore, $\mathbf{z}_t \in \mathbb{R}^{|\mathcal{V}|}$.



From logits to probabilities?

The LM head has now produced one logit for every vocabulary token:

$$\mathbf{z}_t = W_{\text{out}} \mathbf{h}_t + \mathbf{b}.$$

For a tiny toy vocabulary, suppose the resulting logits are

$$z_{\text{days}} = 3.0, \quad z_{\text{week}} = 1.0, \quad z_{\text{approval}} = 0.0.$$

These numbers are **raw scores**. They are not yet probabilities: $3.0 + 1.0 + 0.0 \neq 1$.

Softmax converts all vocabulary logits into a probability distribution:

$$p_{\theta}(y_t = v \mid x, y_{<t}) = \frac{e^{z_{t,v}}}{\sum_{u \in \mathcal{V}} e^{z_{t,u}}}$$

For “days”, $p_{\theta}(y_t = \text{days} \mid x, y_{<t}) = \frac{e^3}{e^3 + e^1 + e^0} \approx 0.844$.



Important Takeaway

The probability comes from relative logits.

It is still only a probability over the next token.



Locally high-probability token can still belong to a false claim

Suppose the generated prefix is:

“Students with a valid medical certificate receive a seven ...”

The model might assign:

Next token	days	day	week	extension	approval
Probability	0.71	0.08	0.07	0.06	0.03

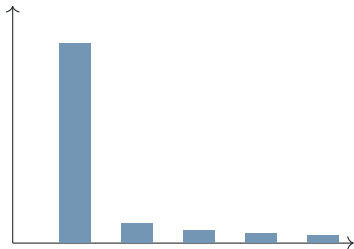
Why can “days” be easy to predict?

- ▶ the phrase “seven days” is *linguistically conventional*;
- ▶ the context already *strongly constrains* the next word;
- ▶ NONE of this checks whether university policy *actually* contains that rule.



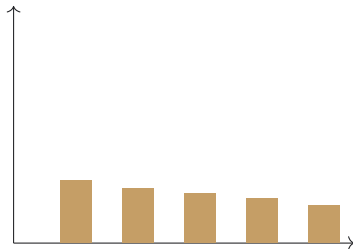
Uncertainty: how *spread* out is the next-token distribution?

Distribution A: peaked



little local ambiguity

Distribution B: spread



many continuations compete



Moral of the story...

We need a number that **summarizes** this spread → **Entropy!**



Entropy, step 1: define the surprise of one event

If an event has probability p , a natural surprise score is

$$I(p) = -\log p.$$

Why this direction?

Probability	Surprise $-\log p$	Intuition
0.90	0.105	expected
0.50	0.693	uncertain
0.10	2.303	surprising

Rare events carry more information because seeing them is more surprising.



Entropy, step 2: average the surprise over all possible next tokens

The next token Y is random according to the model's own distribution $p(y \mid x)$. Expected surprise is

$$\mathbb{E}[I(Y)] = \sum_{y \in \mathcal{V}} p(y \mid x) [-\log p(y \mid x)].$$

Therefore

$$H(Y \mid x) = - \sum_{y \in \mathcal{V}} p(y \mid x) \log p(y \mid x).$$

Entropy is a precise measure of **dispersion of a probability distribution**.



Entropy, step 3: compute two tiny examples

Use natural logarithms.

Peaked distribution

$$p = (0.9, 0.1)$$

$$H = -0.9 \log 0.9 - 0.1 \log 0.1$$

$$H \approx 0.325.$$

Balanced distribution

$$p = (0.5, 0.5)$$

$$H = -0.5 \log 0.5 - 0.5 \log 0.5$$

$$H \approx 0.693.$$

More even probability mass $\Rightarrow$ larger entropy
 $\Rightarrow$ more uncertainty about the next token.

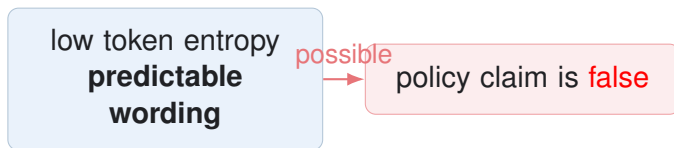


PROBLEM: **Low** token entropy can coexist with a **confidently wrong answer**

Imagine the model has **memorized** or strongly learned the linguistic pattern

“medical certificate” → “seven-day extension”.

Then the token sequence may be extremely predictable.



Token uncertainty $\neq$ actual uncertainty.



Can we combine token probabilities into an **answer-level** score?

For **full** answer $a = (y_1, \dots, y_T)$, the chain rule gives

$$p_{\theta}(a \mid x) = \prod_{t=1}^T p_{\theta}(y_t \mid x, y_{<t}).$$

Taking logs converts the product into a sum:

$$\log p_{\theta}(a \mid x) = \sum_{t=1}^T \log p_{\theta}(y_t \mid x, y_{<t}).$$

Toy example with three generated tokens:

$$0.8 \times 0.7 \times 0.9 = 0.504.$$

NT: Probability of token **sequence $\neq$ a probability that the **answer is correct**.**



Raw sequence probability has a length problem!

Suppose every generated token has probability 0.9.

Answer length	Sequence probability	Same per-token confidence?
$T = 2$	$0.9^2 = 0.81$	yes
$T = 5$	$0.9^5 \approx 0.59$	yes
$T = 10$	$0.9^{10} \approx 0.35$	yes

Longer answers automatically accumulate more factors below 1.

Lower sequence probability even if each token is individually just as predictable.



A common patch: average log-likelihood per token

Raw log-likelihood:

$$\log p(a \mid x) = \sum_{t=1}^T \log p(y_t \mid x, y_{<t}).$$

Normalize by length:

$$\bar{\ell}(a \mid x) = \frac{1}{T} \sum_{t=1}^T \log p(y_t \mid x, y_{<t}).$$

This removes one obvious length effect.

But the question is still: “How probable are the **words** of this answer under the model?” – not “How likely is the **claim** to be correct?”



Closed-choice tasks are easier – because the answer space is explicit

Suppose the question has three options:

- A. Automatic seven-day extension
- B. Instructor approval required
- C. No extension allowed

The model can assign normalized probability mass directly across the choices:

$$p(A \mid x) = 0.20, \quad p(B \mid x) = 0.70, \quad p(C \mid x) = 0.10.$$

Now 0.70 is at least an answer-option probability inside a fixed answer space.

Even here, 0.70 still needs empirical calibration before we interpret it as “70% chance of being correct.”



Part II. Calibrated Confidence



Move from one answer to a population of answers

A single confidence score cannot be audited in isolation. We collect many predictions:

$$(s_1, y_1), \dots, (s_n, y_n),$$

where

- ▶ $s_i \in [0, 1]$ is the model's confidence score;
- ▶ $y_i \in \{0, 1\}$ records whether the answer was **actually correct**.

Toy data:

s_i	.95	.91	.88	.84	.72	.68	.61	.55
y_i	1	0	1	1	0	1	0	1

Calibration is a relationship between scores and outcomes.



Step 1: group predictions with similar confidence

Exact equality $S = p$ is too much of a coincidence with finite data, so we usually create **bins**. Example:

$$B_1 = [0.5, 0.7), \quad B_2 = [0.7, 0.9), \quad B_3 = [0.9, 1.0].$$

For the toy scores:

Bin	Scores inside it
B_1	.68, .61, .55
B_2	.88, .84, .72
B_3	.95, .91

Approximation: “what happens when confidence is **around** p ?”



Step 2: compute empirical accuracy inside one bin

Take

$$B_3 = \{0.95, 0.91\}.$$

Their outcomes are

$$y = \{1, 0\}.$$

So empirical **accuracy** is

$$\text{acc}(B_3) = \frac{1 + 0}{2} = 0.50.$$

Interpretation:

Among these very high-confidence answers, only half were correct.



Step 3: compute average confidence inside the same bin

For

$$B_3 = \{0.95, 0.91\},$$

mean confidence is

$$\text{conf}(B_3) = \frac{0.95 + 0.91}{2} = 0.93.$$

Now compare:

$$\text{conf}(B_3) = 0.93, \quad \text{acc}(B_3) = 0.50.$$

Gap:

$$|0.50 - 0.93| = 0.43.$$

The bin is strongly overconfident.



Repeat the same calculation for every bin

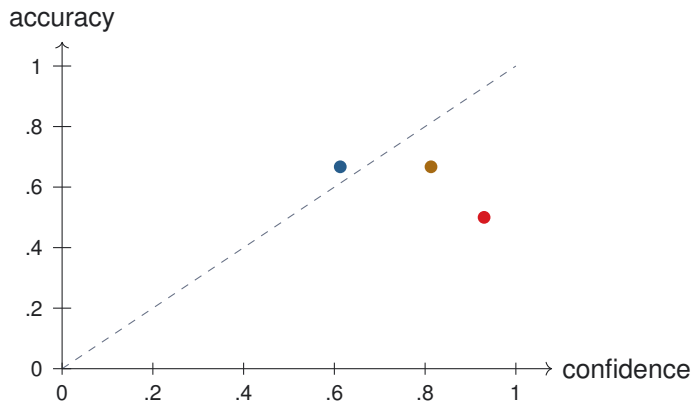
Bin	$ B_m $	$\text{conf}(B_m)$	$\text{acc}(B_m)$
B_1	3	$(.68 + .61 + .55)/3 = 0.613$	$(1 + 0 + 1)/3 = 0.667$
B_2	3	$(.88 + .84 + .72)/3 = 0.813$	$(1 + 1 + 0)/3 = 0.667$
B_3	2	$(.95 + .91)/2 = 0.930$	$(1 + 0)/2 = 0.500$

We get the **calibration pattern**.

Next: **reliability diagram – a visual version of this table.**



Reliability diagram: plot accuracy against confidence



Dashed diagonal = perfect calibration $\implies$ acc = conf.



Reliability diagram $\Rightarrow$ Responsible AI diagnostic

Above diagonal

accuracy $>$
confidence

underconfident

On diagonal

accuracy $\approx$
confidence

well calibrated

Below diagonal

accuracy $<$
confidence

overconfident



Can we compress the whole reliability diagram into one number?

For each bin, compute its **absolute calibration gap**:

$$g_m = |\text{acc}(B_m) - \text{conf}(B_m)|.$$

But bins contain different numbers of examples, so *normalize* each gap by

$$w_m = \frac{|B_m|}{n}.$$

Then add them:

$$\text{ECE} = \sum_{m=1}^M \frac{|B_m|}{n} |\text{acc}(B_m) - \text{conf}(B_m)|.$$

ECE is a weighted average of the vertical gaps in the reliability diagram.



Compute ECE for our toy example – slowly

We had $n = 8$.

Bin	Weight	Accuracy	Confidence	Gap
B_1	3/8	.667	.613	.054
B_2	3/8	.667	.813	.146
B_3	2/8	.500	.930	.430

Therefore

$$\text{ECE} = \frac{3}{8}(0.054) + \frac{3}{8}(0.146) + \frac{2}{8}(0.430) \approx 0.183.$$

Average calibration mismatch is about 18.3 percentage points for this toy sample.



ECE is useful – but **NOT** perfect metric

ECE depends on **choices** we made:

- ▶ number and boundaries of bins;
- ▶ finite-sample **noise** inside each bin;
- ▶ **averaging** can hide severe local miscalibration;
- ▶ *two models can have similar ECE but very different individual errors.*

Can we score every probabilistic prediction **directly**, without binning?

Brier score in rescue!



Brier score, step 1: score one probabilistic prediction

For binary correctness,

$$y \in \{0, 1\}, \quad s \in [0, 1].$$

Use squared error:

$$L_{\text{Brier}}(s, y) = (s - y)^2.$$

Two wrong predictions ($y = 0$):

Confidence s	Loss $(s - 0)^2$	Interpretation
0.55	0.303	cautiously wrong
0.95	0.903	confidently wrong

Brier score punishes confident mistakes much more strongly.



Brier score, step 2: average across examples

For n examples,

$$\text{Brier} = \frac{1}{n} \sum_{i=1}^n (s_i - y_i)^2.$$

Unlike ECE:

- ▶ every prediction contributes directly;
- ▶ no confidence bins are required;
- ▶ both calibration and sharpness affect the score.

Smaller is better; 0 is perfect.



NT: Truthful probability reporting optimal under Brier loss

Suppose the true chance of correctness is q but the model reports s . Expected Brier loss is

$$\mathbb{E}[(s - Y)^2] = q(s - 1)^2 + (1 - q)s^2.$$

Expand:

$$= q(s^2 - 2s + 1) + (1 - q)s^2 = s^2 - 2qs + q.$$

Differentiate with respect to s :

$$\frac{d}{ds} = 2s - 2q.$$

Set to zero:

$$s = q.$$

The expected Brier loss is minimized by reporting the true probability – A proper scoring rule.



Part IV. Rectifying Miscalibration



Suppose the model ranks answers well but too sharply

Imagine a classifier-like LLM head that often chooses the **correct option**, but says things like

0.99, $\rightarrow$ **0.97**, $\rightarrow$ **0.95**

when empirical accuracy is much **lower**.

We want to keep the ranking of logits but **soften the probability distribution**.

Temperature scaling: change confidence without changing logit rank.



Temperature scaling starts from the same logits

Original softmax:

$$p_i = \frac{e^{z_i}}{\sum_j e^{z_j}}.$$

Introduce temperature $T > 0$:

$$p_i^{(T)} = \frac{e^{z_i/T}}{\sum_j e^{z_j/T}}.$$

Interpretation:

- ▶ $T = 1$: original probabilities;
- ▶ $T > 1$: logits are divided by a larger number $\Rightarrow$ probabilities become softer;
- ▶ $T < 1$: probabilities become sharper.



A numerical example: same logits, different temperature

Take logits

$$z = (3, 1, 0).$$

At $T = 1$:

$$p^{(1)} \approx (0.844, 0.114, 0.042).$$

At $T = 2$:

$$z/T = (1.5, 0.5, 0),$$

so

$$p^{(2)} \approx (0.629, 0.231, 0.140).$$

The top class stays the same, but confidence drops from 0.844 to 0.629.

Temperature changes probability sharpness, not the ordering of logits.



How do we choose the temperature?

Do **not** look at the test set.

Use a held-out calibration set (x_i, y_i) and choose T to minimize negative log-likelihood:

$$T^* = \arg \min_{T > 0} - \sum_{i=1}^n \log p_T(y_i | x_i).$$

Pipeline:

1. **Freeze** the trained model.
2. Collect logits on validation examples.
3. Search for one scalar T .
4. Apply that T at deployment time.

Temperature scaling is **post-hoc** calibration.



What temperature scaling can fix – and what it cannot

Can help with	Cannot guarantee
systematic over/under-confidence ✓	factual correctness ✗
probability sharpness ✓	evidence grounding ✗
validation-domain calibration ✓	calibration under arbitrary domain shift ✗
<u>closed-choice</u> confidence ✓	trustworthy free-form answer probability ✗

Calibration methods **repair a mapping** from scores to frequencies;
Cannot create missing knowledge / evidence.



Part V. Self-awareness



Verbalized 95% confidence – can self-evaluation be made more structured?

Instead of asking only:

“How confident are you?”

We can ask the model to evaluate a **specific proposition**: Example proposition:

“Students with a valid medical certificate are automatically entitled to a seven-day extension.”

Then ask:

“What is the probability that this statement is true?”



Pushing the LLM to think on a "**claim**" (rather than confidence)

From generic confidence toward a better-defined event: *is this **claim** true?*



$P(\text{True})$: probability assigned to the truth of a candidate answer

Conceptually, suppose the model is asked to choose between

True and False

for a **proposed answer a** . Then define

$$P(\text{True} \mid x, a)$$

as the model's normalized probability on "True." Example:

$$P(\text{True} \mid x, a) = 0.72.$$

More explicit than "I feel 72% confident" – *still a model-produced score that must be validated*.



Pushing the LLM to think on a "**self-knowledge**" (rather than claim)

From confidence on claim toward
self-knowledge: *Am I even **capable** to answer?*



$P(\text{IK})$: can the model predict whether it knows before answering?

Say we elicit instead:

“Do I have enough internal knowledge to answer this question correctly?”

Let

$$P(\text{IK} \mid x)$$

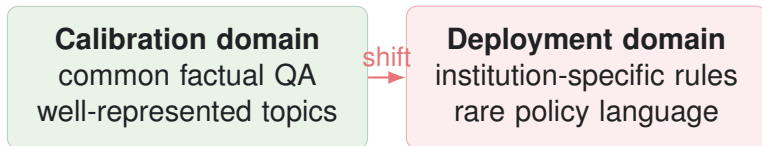
represent the model's estimate that it **knows** the answer.

$P(\text{True})$	$P(\text{IK})$
score a particular proposed answer	score ability before committing to an answer
<i>“Is this answer true?”</i>	<i>“Do I know enough to answer?”</i>



But self-knowledge can fail under domain shift

Suppose calibration was measured on ordinary factual QA, but deployment is on IIT Delhi policy documents.



Calibration can degrade because the relationship between score and correctness may change.

A confidence model is only as trustworthy as the conditions under which its calibration remains valid.



A small thought experiment: two assistants with the same 80% confidence

Assistant A

80% confidence
calibrated on the same policy domain
retrieved current policy

Assistant B

80% confidence
calibrated on generic QA
answering from parametric memory

Same number. Very different reasons to trust it.

Responsible use of confidence requires context about how the score was produced and validated.



Part VI. Utility



Decision

Suppose we have a confidence score $S(x, a)$ that has been validated for our task. A naive rule is

$$\pi(x, a) = \begin{cases} \text{answer}, & S(x, a) \geq \tau, \\ \text{do not answer}, & S(x, a) < \tau. \end{cases}$$

Two questions:

1. How should we choose the threshold τ ?
2. What should “do not answer” actually mean: *abstain, retrieve, ask, or hand off*?

Lecture 3: Selective generation and abstention.



Confidence alone should not determine action in a responsible system

For the university policy assistant:

Confidence	Context	Sensible action
high	current policy retrieved	answer with citation
high	no evidence, institution-specific rule	retrieve before answering
low	question is ambiguous	ask clarification
low	relevant evidence may exist	retrieve more
low	high-stakes and unresolved	abstain / human handoff

Confidence $\neq$ evidence, task risk, or accountability.



So far...

We now understand	Still unresolved
where confidence scores can come from	whether repeated generations disagree in meaning
token probability and entropy	whether individual claims are supported
calibration and reliability diagrams	when the system should abstain
ECE and Brier score	whether retrieved evidence grounds the answer
temperature scaling	judge / verifier / agent uncertainty
structured self-evaluation	system-level auditability



One final check before we leave

Suppose tomorrow the same assistant says:

“I am 95% confident that this university policy answer is correct.”

What should you ask before trusting the number?

1. **Where did 95% come from?** verbalized confidence, logits, judge, verifier, something else?
2. **What event is it supposed to predict?** token choice, answer correctness, factual support?
3. **Was this score calibrated?** on what data and what domain?
4. **What should happen when confidence is low – or evidence is missing?**

If unanswered, **the 95% should not carry much authority.**



Lecture 1 takeaway

Confidence is not trust.

Trust begins when a score has a clear provenance, a clear target event, empirical calibration, and an appropriate action policy.

Next:

Uncertainty Beyond Confidence – From Repeated Generations to Claims