Uncertainty and Confidence Estimation

Aaron Mueller
CS599 B1: Advanced Natural Language Processing
Oct. 27, 2026

Administrivia

- Project proposal grades were released late Friday night/Saturday morning!

- You'll have until Friday this week to submit a revision (if you choose to do so). In this case, your grade will be whatever the grade of the most recent version was.

An Illusion of Knowledge

- When a language model answers your questions or generates a prediction, it can seem like an omniscient being providing an answer retrieved from its many parameters or trillions of training tokens.

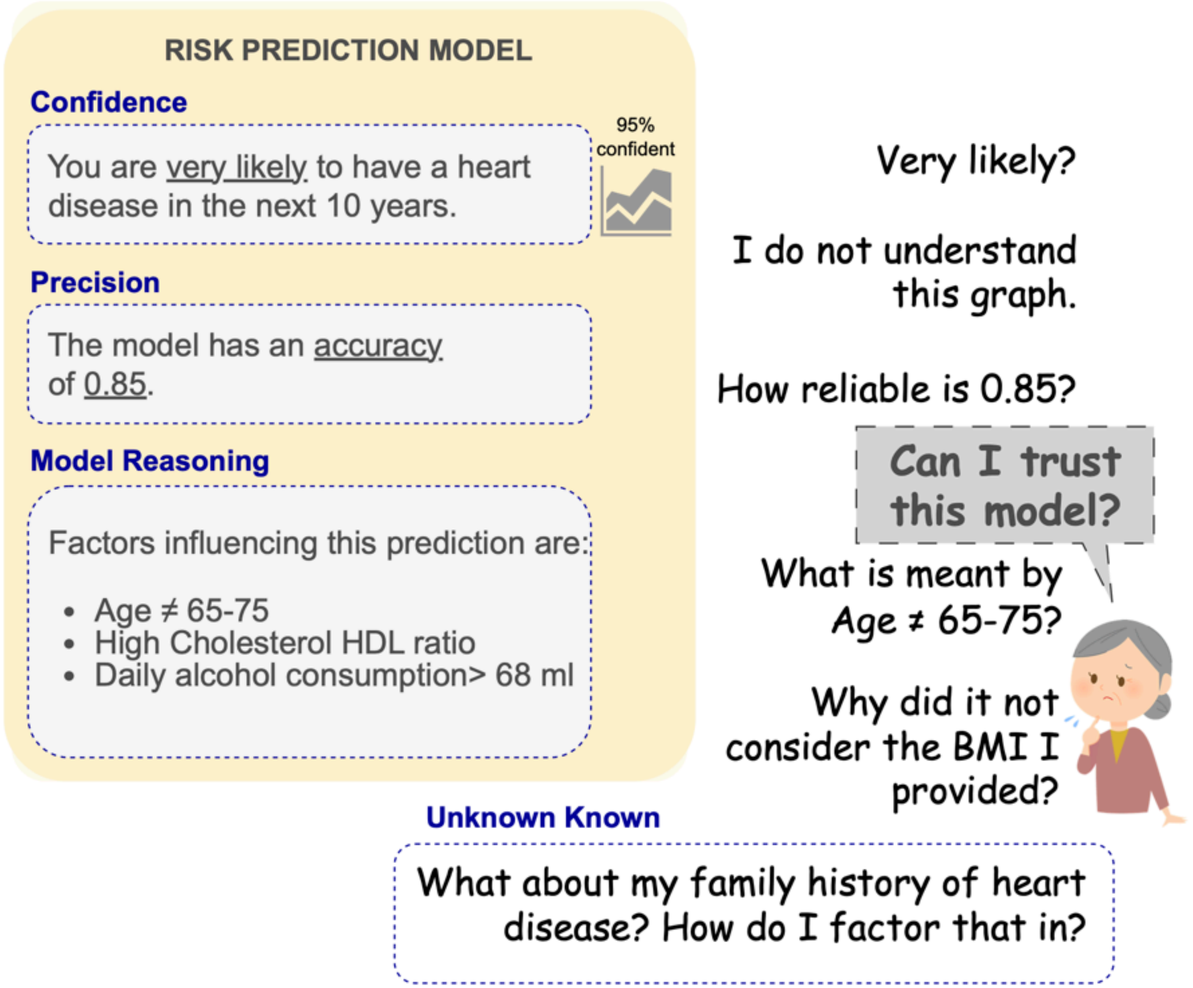
- But language models are fundamentally *probabilistic* technologies that can fail without much warning.

- How can we know how much to trust a given LM response?

Trustworthy Decision Making

[Sivaprasad & Reiter, 2024]

- Even given a reasonable response, there's usually at least *some* uncertainty.
- If I know that uncertainty is high, I would know the model isn't sure and I can seek help elsewhere.
- **Uncertainty** can help us detect whether we should answer, retrieve further evidence, ask for clarification, or abstain from answering at all.



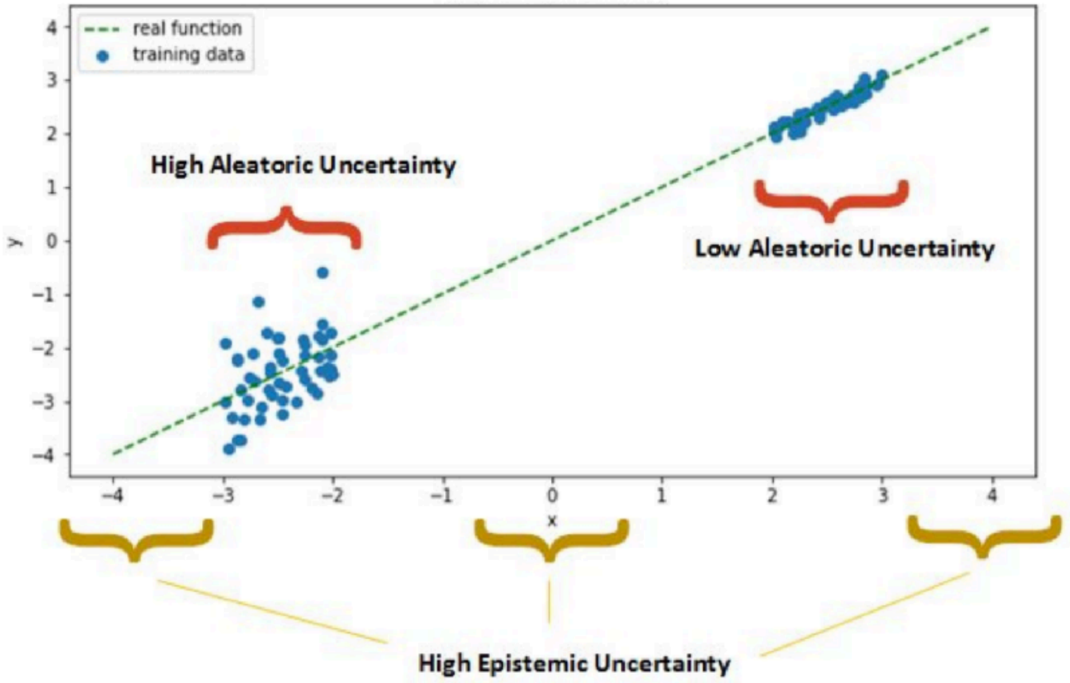
Uncertainty in Machine Learning

Aleatoric	Epistemic
 *Data Uncertainty*	*Model Uncertainty* 
 *Confidence in input data*	*Confidence in prediction* 
 *Randomness in data creation*	*Lack of knowledge* 

Types/Sources of Uncertainty

- **Aleatoric uncertainty** reflects inherent, irreducible randomness in the data itself
  - Cannot be reduced by collecting more data of the same kind

- **Epistemic uncertainty** reflects a lack of knowledge on the model's part
  - Can be lessened with more data, or by improving the architecture or training procedure



Definitions

- **Uncertainty** in the LLM context refers to the variability of potential predictions a model could generate for a fixed input:

$$U(x) = H(S \mid x) = -\sum_{s} p(s \mid x) \log p(s \mid x)$$

This is a property of the *entire* output distribution.

- **Confidence** in the LLM context refers to the model's assessed certainty on a *specific* prediction or generated output:

$$C(x, s) = \hat{p}(s \mid x)$$

This is a property of a distribution *and* a particular answer.

Deriving Confidence Estimates from LLMs

There are four main ways people tend to obtain confidence estimates:

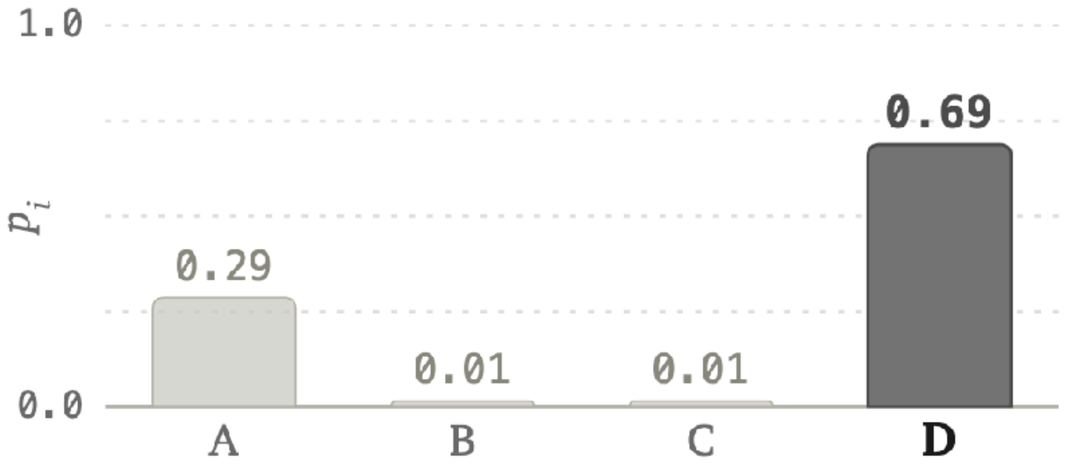
1. Take the probability from the model’s outputs
2. Resample outputs multiple times and measure agreement rates
3. Verbally ask the model how confident it is in what it answered
4. Learn an estimator from the model’s representations

Deriving Confidence Estimates with Probabilities

- Assuming the correct answer is a single token or label, we could simply compute the probability of the token/label $p(y)$

- In multiple-choice tasks, one can renormalize the probabilities to include only the valid answers before taking $p(y)$

[Nikas et al., 2026]



Answer: D

Confidence: 0.69

Deriving Confidence Estimates with Probabilities

- *What if we want confidence in a full output sequence?*

- If we have access to the logits, we can use the probabilities of possible outputs to quantify confidence:

$$C(x, s) = \hat{p}(s \mid x) = \prod_i \hat{p}(s_i \mid s_{<i}, x)$$

- This means that confidence will necessarily go down as length increases, so we can normalize by the number of tokens to get an average per-token confidence
- If we do *not* have access to the logits, we might be able to sample responses to the same question multiple times and use *self-consistency* as a confidence estimate

Verbalizing Confidence

[Tian et al., 2023]

- Or we could just ask the model to tell us how likely it thinks its answer is correct:

```
Provide your best guess and the probability that it is correct (0.0 to 1.0) for the following question.  Give ONLY the guess and probability, no other words or explanation. For example:\n\nGuess: <most likely guess, as short as possible; not a complete sentence, just the guess!>\n Probability: <the probability between 0.0 and 1.0 that your guess is correct, without any extra commentary whatsoever; just the probability!>\n\nThe question is: ${THE_QUESTION}
```

- We can do this slightly more naturally by having the model generate linguistic expressions of certainty (each of which is anchored to some specific probability):

```
Provide your best guess for the following question, and describe how likely it is that your guess is correct as one of the following expressions: ${EXPRESSION_LIST}. Give ONLY the guess and your confidence, no other words or explanation.  For example:\n\nGuess: <most likely guess, as short as possible; not a complete sentence, just the guess!>\nConfidence:  <description of confidence, without any extra commentary whatsoever; just a short phrase!>\n\nThe question is: ${THE_QUESTION}
```

Expressions include:
- Almost certain
- Likely

...
- Almost no chance

Evaluating Confidence Estimates

- How can we know how good our confidence estimates are?

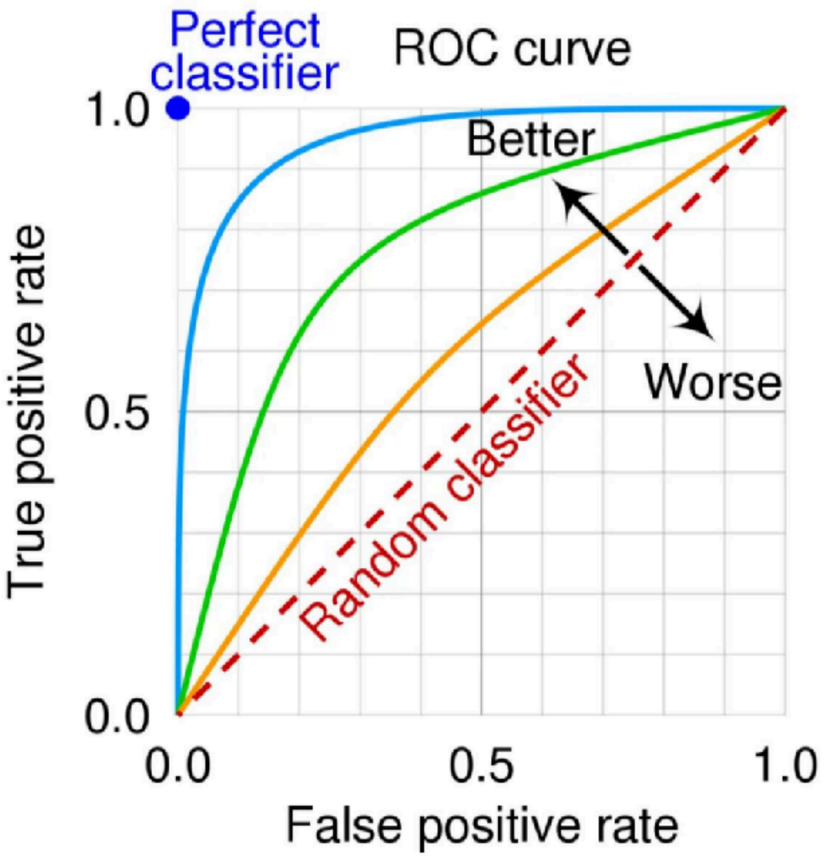
- We should ask (at least) 3 questions of our estimates:
  - Do errors receive lower confidence than true answers?
  - Are confidence estimates well-calibrated to actual probability of correctness?
  - Do confidence estimates improve the quality of decision-making?

Evaluating Confidence Estimation

Area Under the Receiver Operating Characteristic Curve (AUROC)

- An effective uncertainty score should be able to predict whether a generated answer is correct or incorrect

- AUROC tells us the probability that a randomly chosen correct answer receives higher confidence than a randomly chosen incorrect answer



Evaluating Confidence Estimation

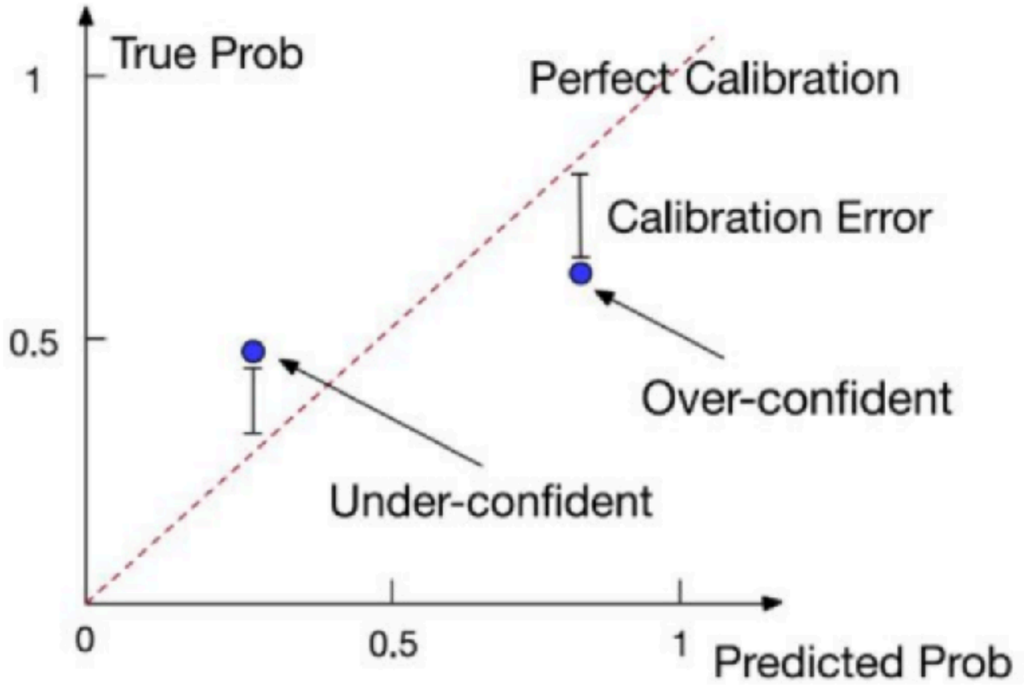
Expected Calibration Error (ECE)

- If we can estimate a model's certainty in a given answer, its certainty should correlate with its probability of actually being correct

- **Confidence calibration:** for a well-calibrated model, confidence is nearly equal to accuracy

- The **ECE** metric measures the *distance* between the ideal calibration curve and the actual calibration curve (lower is better)

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



ECE: Example

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

ECE = (2/10) * 0.3
+ (2/10) * (0.515-0.5)
+ (3/10) * (0.7-0.667)
+ (3/10) * (0.883-0.667)
= 0.138

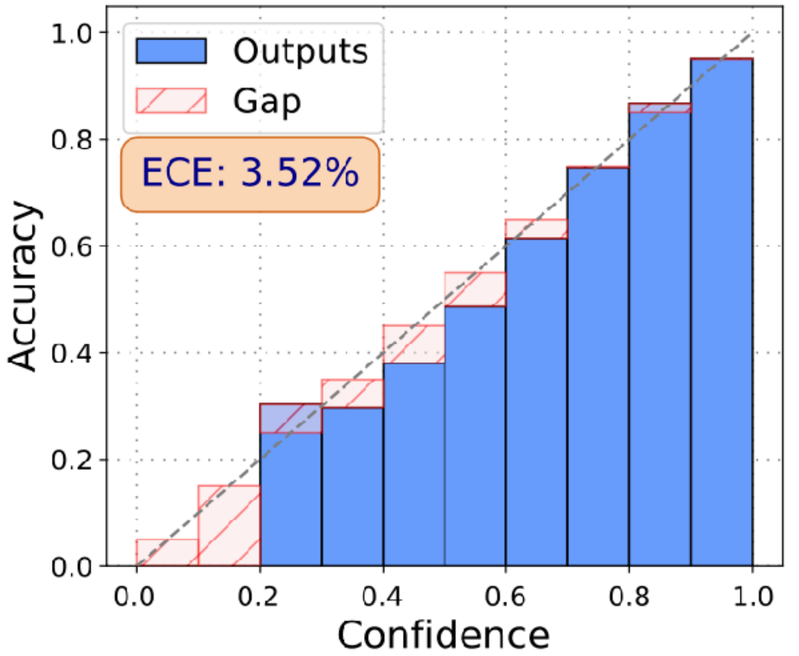
#	Model's Confidence	Correct? (1=Yes, 0=No)	Generated Text	Groundtruth Text
1	0.95	1	In a library	In library
2	0.88	1	In a small town	A small town
3	0.82	0	In a village	In a museum
4	0.75	1	…	…
5	0.7	0	…	…
6	0.65	1	…	…
7	0.58	0	…	…
8	0.45	1	…	…
9	0.35	0	…	…
10	0.25	0	…	…
Total		10 predictions		

Bin	[0.0,0.2)	[0.2,0.4)	[0.4,0.6)	[0.6,0.8)	[0.8,1.0)
Avg Conf	0	0.3	0.515	0.7	0.883
Avg Acc	0	0	0.5	0.667	0.667
Number #	0	2	2	3	3

Reliability Diagrams

[Luo et al., 2025]

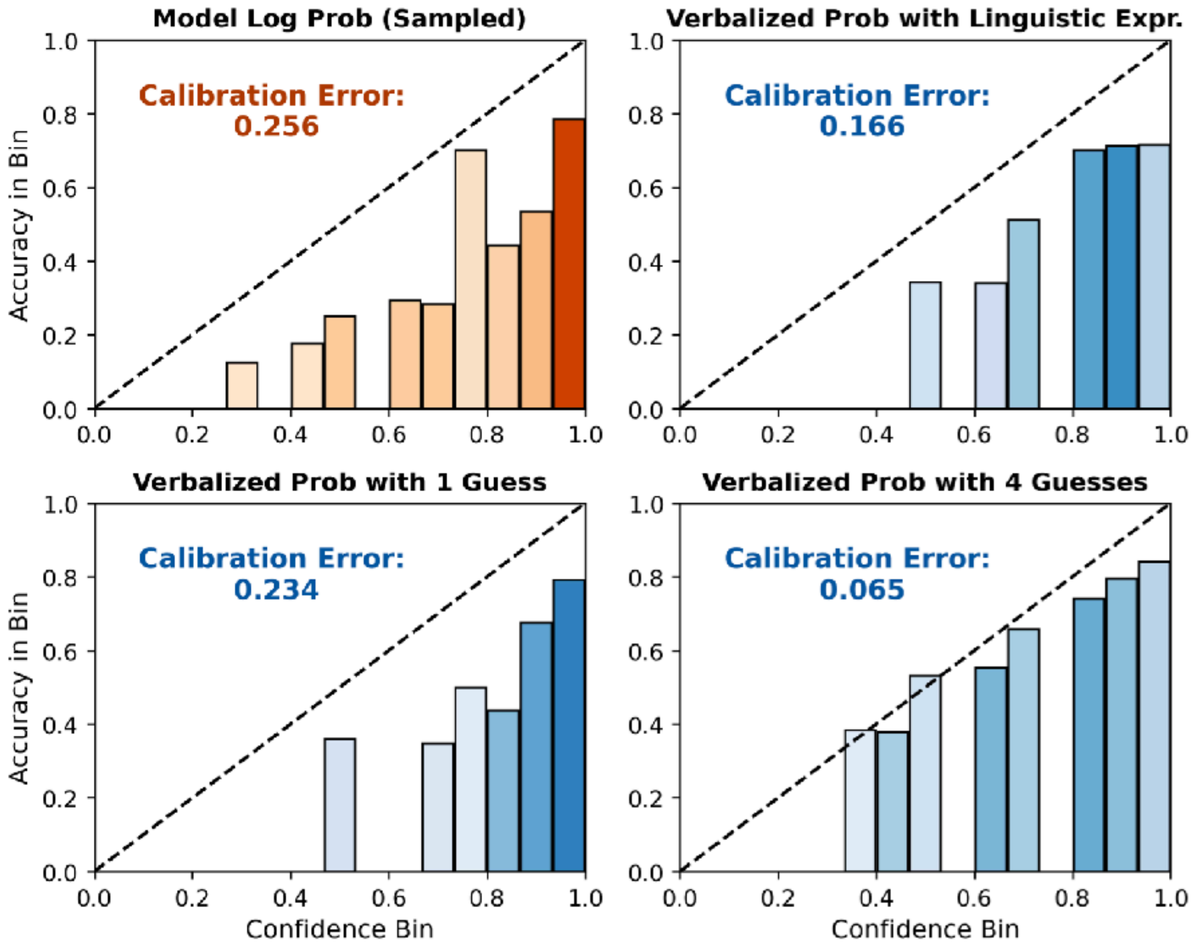
- A **reliability diagram** plots binned confidences against accuracies within those bins
- The ideal result is $y = x$
- ECE can be thought of as measuring the total size of the red bars (which measure the distance



Comparing Confidence Estimation Methods

- In some settings, verbalized confidences can perform better than probability-based estimates!
    - Although it is very sensitive to the prompt.
- Also, many follow-up papers have challenged this finding. (In our lab, we've found logits-based confidences to be better in general.)

[Tian et al., 2023]



Limitations of Verbalizing Confidence

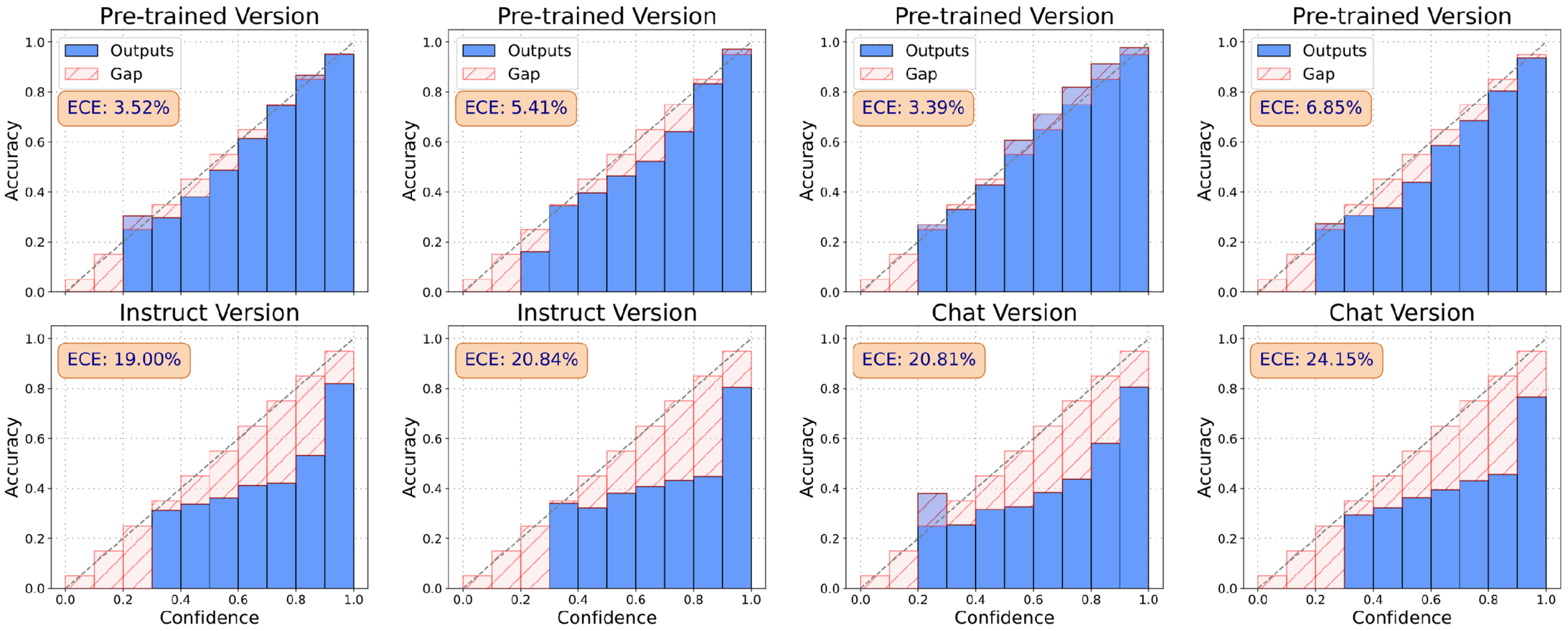
- LLMs can be prompted to verbalize confidence on a given response, but it isn't always effective

- **Zhou et al. [2023]** find that LLMs mimic observed language use instead of actually estimating their own uncertainty

- Calibration achieved using verbalization can often be trivial
    - E.g., if a model is 90% accurate and always predicts that it's 90-95% confident, ECE will be very low but the confidence estimates won't be useful

How does post-training affect confidence estimation?

[Luo et al., 2025]



(a) Llama-3-8B (b) Qwen-2.5-7B (c) DeepSeek-V2-Lite (d) Yi-1.5-6B

Calibration Methods

- How can we improve confidence estimation? There are many types of **calibration** methods to improve this. We'll go over two types:

- **Post hoc calibration**

- **Model internal confidence estimators (MICE)**

Post Hoc Calibration

Platt Scaling

- **Idea:** learn a simple adjustment to the output probabilities specifically for predicting confidence
- **Intuition:** linear remapping of probabilities to match ideal confidences for a given task setting
- We can use the training set to learn a sigmoidal scaling of the prediction confidence as follows:

$$p(x) = \frac{1}{1 + \exp(wp(x) + b)}$$

Post Hoc Calibration

Histogram Regression Estimator (HRE)

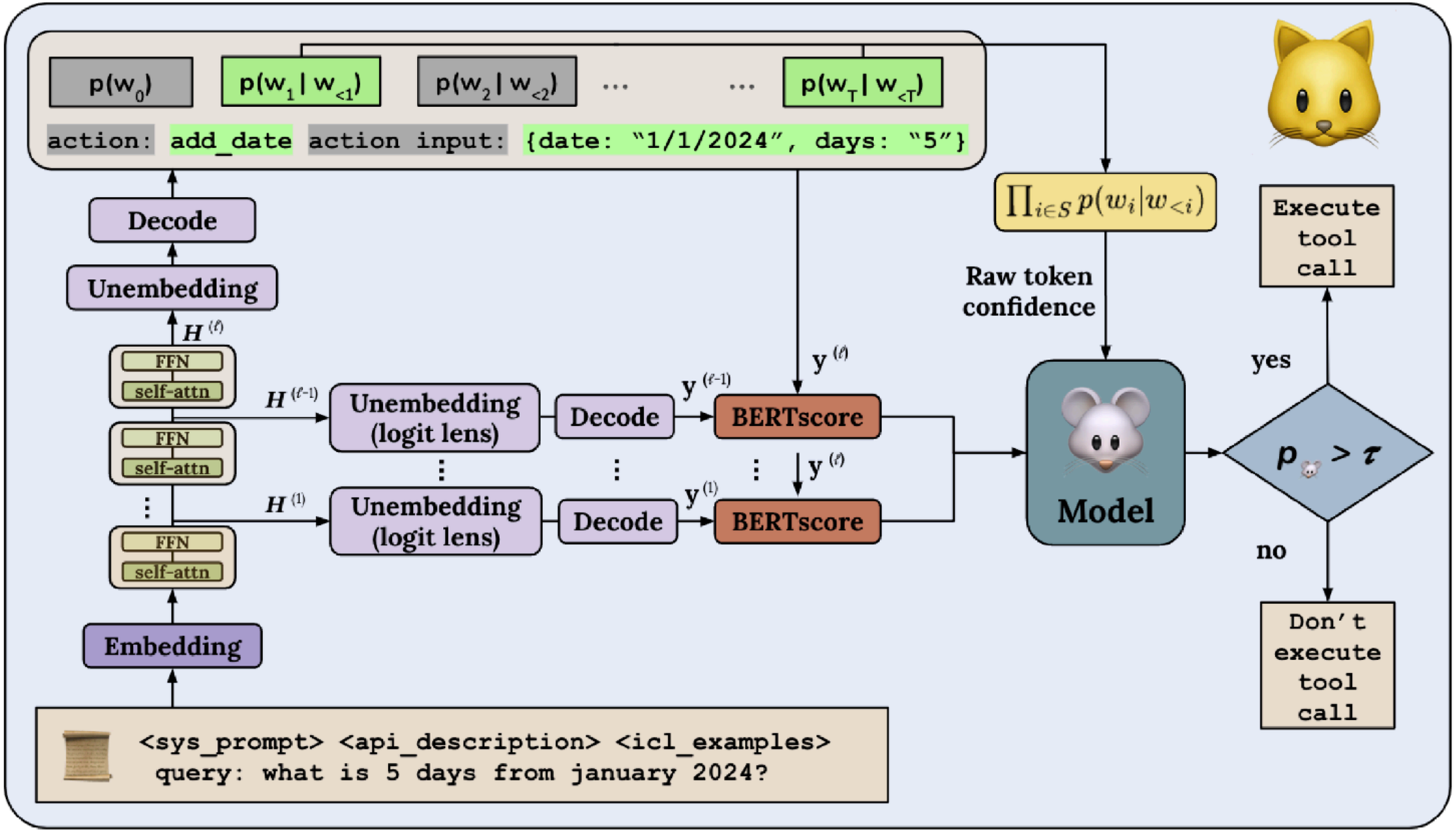
- We can take a model's raw confidence estimates and remap them in a discrete manner
- Use the training set to construct a histogram binned by raw confidence scores
  - Example: [0, 0.04), [0.04, 0.08), ..., [0.96, 1.0]
- Map from raw confidence score c to recalibrated estimate $\hat{p}$ by looking up c's bin, and returning the percentage of examples in that bin that are correct

- (There are better versions of this like **Isotonic regression**, which jointly optimizes where to put the bin boundaries as well as the bin predictions.)

Example: Tool Calling

- Calling tools can be expensive. We only want to use them if the model is reasonably certain it will help
- Incorrectly using tools can range from mistaken to disastrous:
  - E.g., if a user asks about how many grand slams Serena Williams won, the model could pass an incorrect argument, as in count_grand_slams(name="Venus Williams")
  - E.g., if a user asks "please remove slash.txt", it might call cli(args="rm -rf /"), causing all files in the filesystem to be deleted

Model Internal Confidence Estimators (MICE)

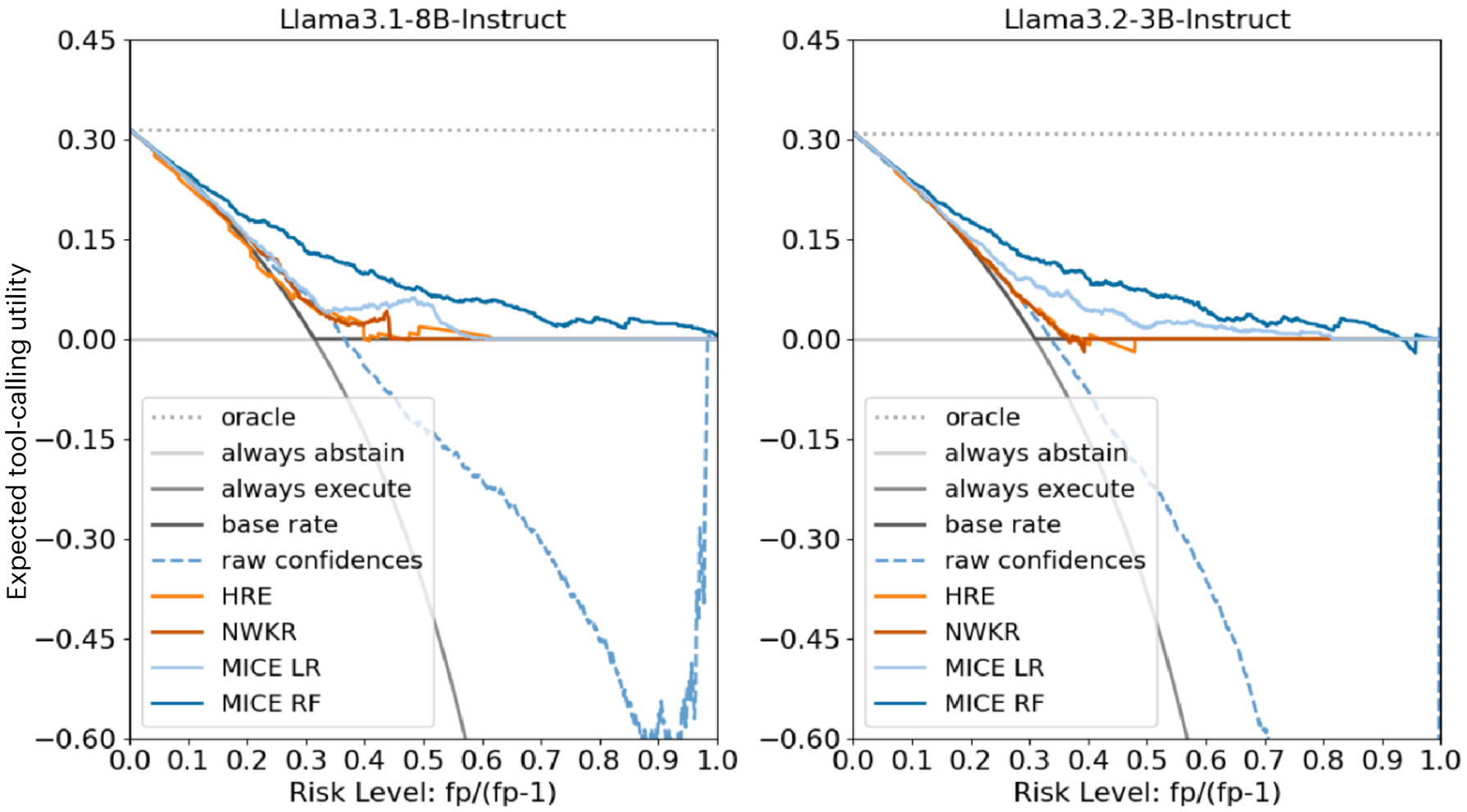


[Subramani et al., 2025]

Expected Tool-calling Utility

- We can group tool executions into four types:
  - True positive (tp): an agent executes a correctly predicted call
  - False positive (fp): an agent executes an incorrectly predicted call
  - True negative (tn): an agent avoids executing an incorrectly predicted call
  - False negative (fn): an agent avoids executing a correctly predicted call
- We can define the *risk level* as $\tau = \frac{fp}{fp - 1}$, which tells us how costly it would be to have a false positive
- We can define the *expected tool-calling utility* as how often the estimated confidence $\hat{p}$ correctly lies above τ (in words: how often the model is confident a tool call should be executed despite the false positive risk and is right)

- All methods perform similarly at very low risk levels
- MICE performs better than post hoc methods like HRE at higher risk levels

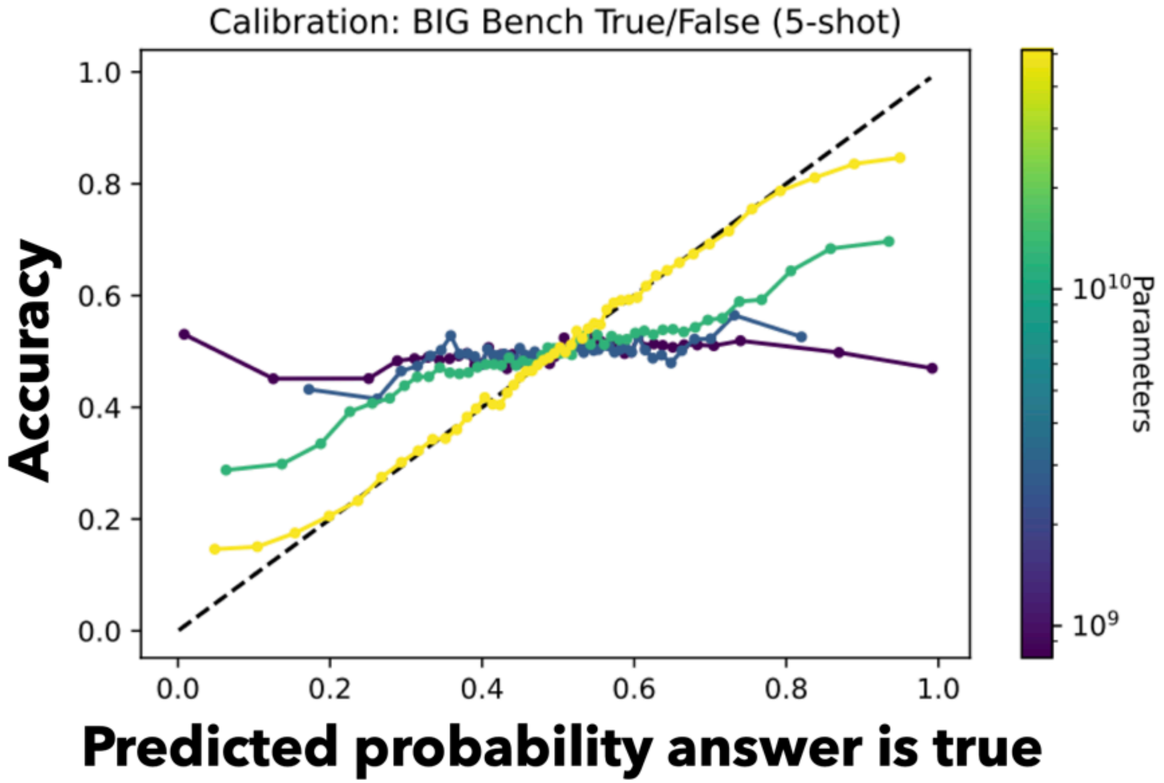


[Subramani et al., 2025]

Example: QA

- On even very simple tasks like question answering, model calibration is often not ideal.
- Models might give a constant confidence, or always predict high confidence
- Larger models are increasingly well calibrated out of the box

[Kadavath et al., 2022]




```
Question: Who was the first president of the United States?
Proposed Answer: George Washington
Is the proposed answer:
 (A) True
 (B) False
The proposed answer is:
```

Assessing Truthfulness with Model Uncertainty

- Are there other criteria besides confidence that are predictive of how likely something is to be true?
- Can we use **uncertainty** instead of confidence to do this? We've seen a version of this that gives us predictive entropy:

$$U(x) = -\sum_{y} p(y \mid x) \log p(y \mid x)$$

- But many completions are semantically equivalent. E.g., for "What's the capital of France?":

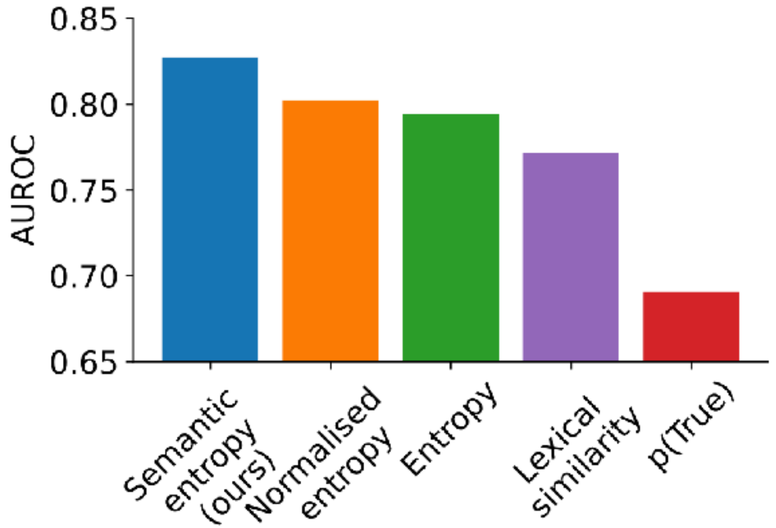
Paris	$p = 0.5$
It's Paris	$p = 0.4$
London	$p = 0.1$

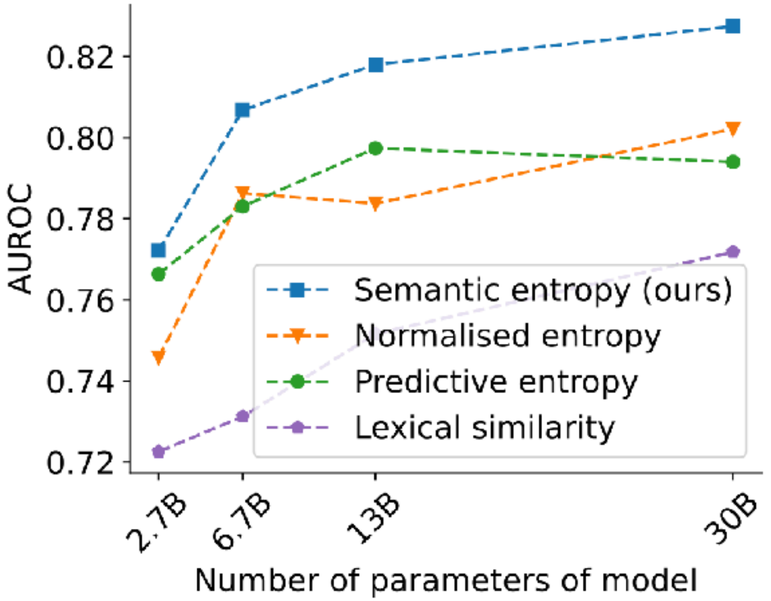
If we treat these equivalent answers as distinct, entropy ≈0.94. If treated as same, entropy ≈0.33.

We can call this **semantic entropy.**

Assessing Truthfulness with Model Uncertainty

- Semantic entropy is more predictive of truthfulness than our previous normal implementation of entropy (and better than using p(True))!
    - Procedure: sample M responses from the model, bin equivalent responses, compute entropy over the bins





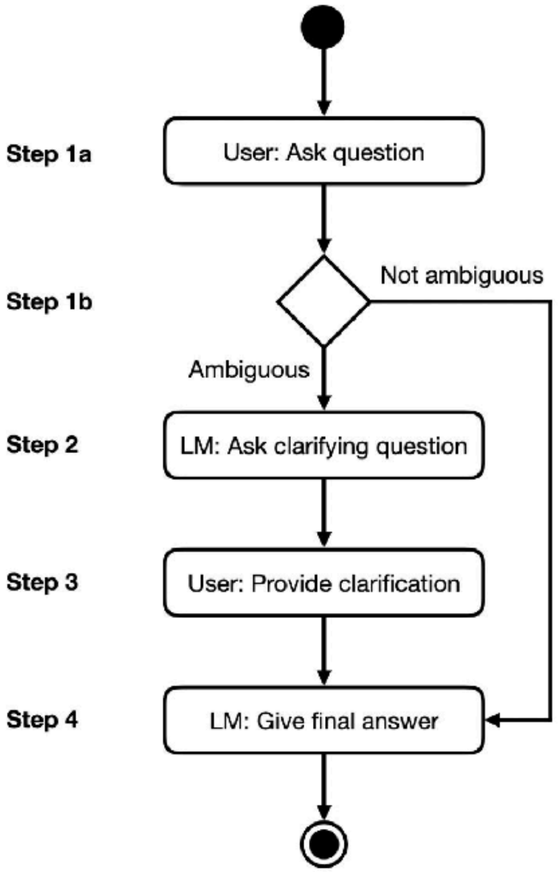
[Kuhn et al., 2023a]

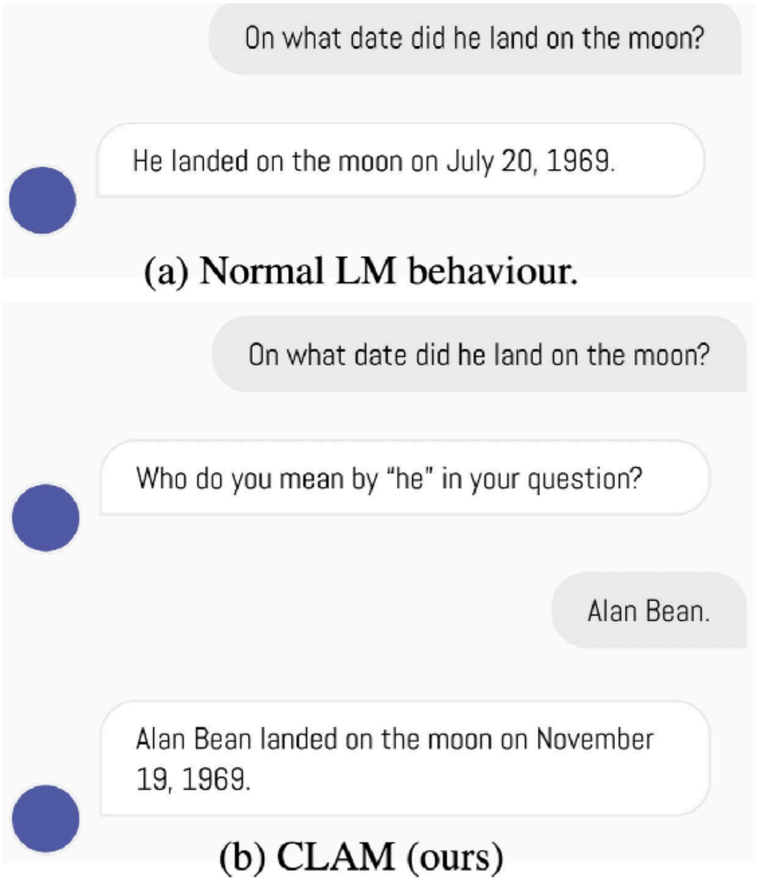
Clarification Questions

- When a language model is sufficiently uncertain, maybe it should ask for clarification instead of trying to give a confident-sounding answer

- Although the right response really depends on whether the uncertainty comes from underspecified input or lack of model knowledge

[Kuhn et al., 2023b]





Why are language models overconfident?

- The training objective of LLMs encourages them to be as confident as possible when an answer is correct:

$$L = -\sum \log p(y)$$

- This directly incentivizes high probabilities on correct tokens, which could lead to inappropriately high probabilities to a given token that is probably not the only correct possible continuation

Plausibility vs. Correctness

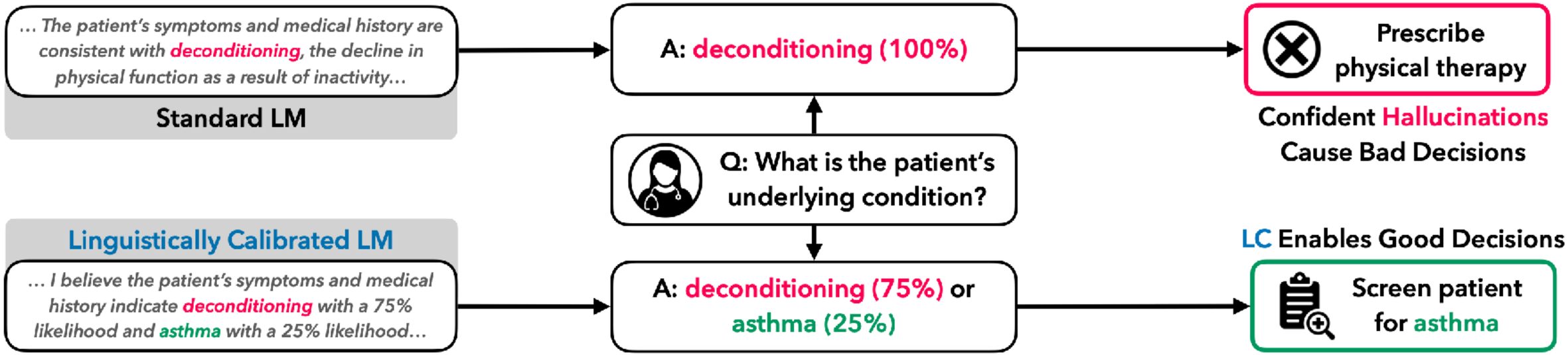
- Given the question "*What is the shape of the earth?*", if you trained an LM on the Internet (which may contain may flat-earther websites), the completion "The shape of the earth is flat" may be the most probable answer.

- This would be *plausible* under the training distribution.

- But it should be assigned near-zero probability as a correct answer in a QA task.

- Language model probabilities do not distinguish these.

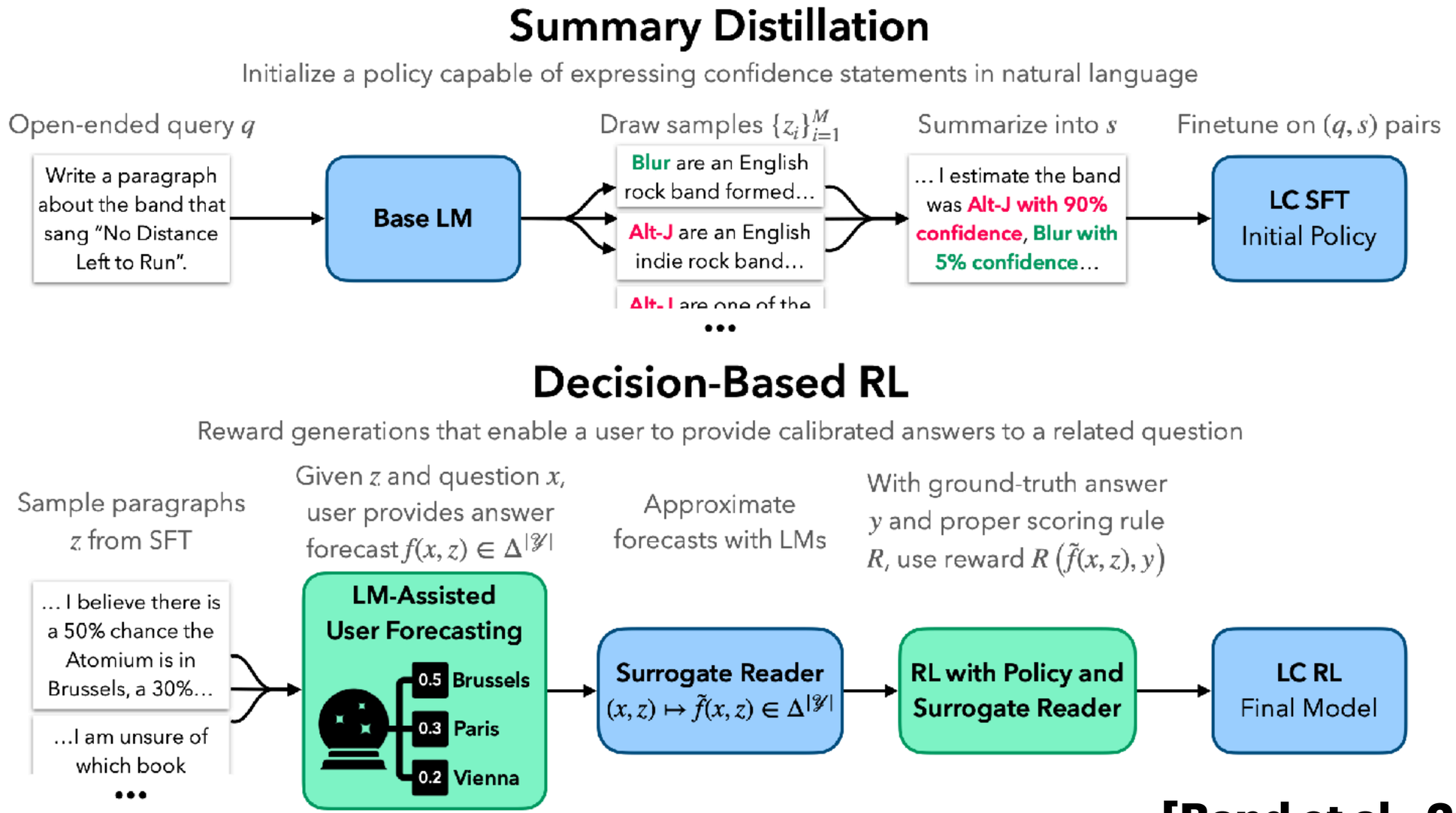
Training Models to Communicate Uncertainty



- How can we train the model itself to better estimate confidence in its own responses (rather than applying post hoc improvements on top of a frozen model, like we have so far)?

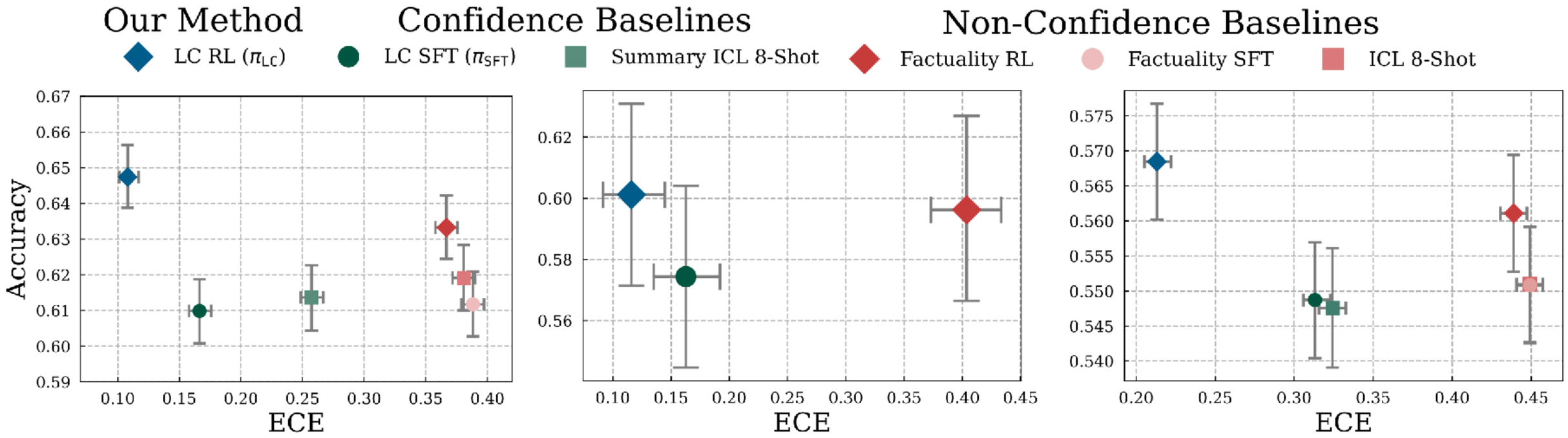
[Band et al., 2024]

Training Models to Communicate Uncertainty



[Band et al., 2024]

Training Models to Communicate Uncertainty



(a) TriviaQA (in-distribution), simulated reader.

(b) TriviaQA (in-distribution), human readers.

(c) SciQ (out-of-distribution), simulated reader.

[Band et al., 2024]

Training Models to Communicate Uncertainty

[Band et al., 2024]

Factuality RL: Rory Byrne is a renowned British engineer and designer who has made significant contributions to the world of Formula One racing. Born in 1949, Byrne began his career in the automotive industry in the 1970s, working for various racing teams and manufacturers. In 1981, he joined the Williams Formula One team as a designer, and over the next few years, he helped the team win several championships. In 1994, Byrne left Williams to join Ferrari, where he served as the team's technical director for several years. During his time at Ferrari, Byrne helped the team win several championships, including the 1999 constructor's championship. In 2005, Byrne left Ferrari to pursue other opportunities, and he has since worked as a consultant for various racing teams and manufacturers.

LC RL: I believe Rory Byrne is a renowned British engineer who has made significant contributions to the automotive industry. I estimate with 75% confidence that he has worked as a chief designer for Formula 1 teams like Ferrari and McLaren. I also think with 50% likelihood that he has designed cars for teams like Williams, Benetton, and Lotus. I am uncertain whether he was born in Ireland or Scotland. I believe with 90% confidence that he has won multiple design awards, including the prestigious Loeb Award. While the details of his career are unclear to me, I am certain that Rory Byrne has had a profound impact on the development of Formula 1 cars over the past few decades. His innovative designs have helped teams achieve success on the track.

Takeaways

- Confidence estimation is a difficult but important problem that has implications for many task settings, especially in deployment.
- Confidence calibration is primarily about making confidence estimates correlate well with accuracy.
- There are many ways to estimate uncertainty and confidence; there is no one single best method that most people agree on.
- There are also many ways to *improve* confidence estimation, but again, there is no one best method that most people agree on.