Foundation Models That Can Tell Us When They Don't Know

Yarin Gal

yarin@cs.ox.ac.uk

In today's talk

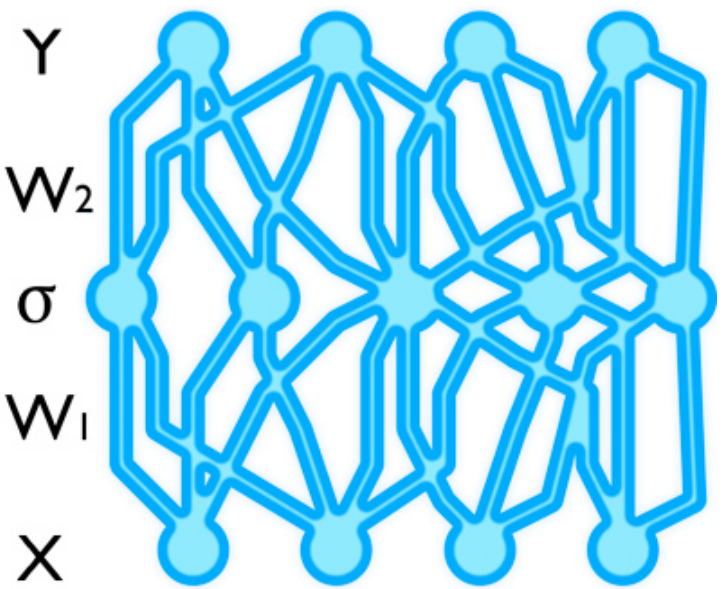
UNIVERSITY OF OXFORD

Conceptually simple models

Data: $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$, $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_N\}$
Model: given matrices $\mathbf{W}$ and non-linear func. $\sigma(\cdot)$, define "network"

$$\tilde{\mathbf{y}}_i(\mathbf{x}_i) = \mathbf{W}_2 \cdot \sigma\left(\mathbf{W}_1 \mathbf{x}_i\right)$$

Objective: find $\mathbf{W}$ for which $\tilde{\mathbf{y}}_i(\mathbf{x}_i)$ is close to $\mathbf{y}_i$ for all $i \leq N$.



Pillar I: Deep learning

UNIVERSITY OF OXFORD

Conceptually simple models

Data: $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$, $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_N\}$
Model: given matrices $\mathbf{W}$ and non-linear func. $\sigma(\cdot)$, define "network"

$$\tilde{\mathbf{y}}_i(\mathbf{x}_i) = \mathbf{W}_2 \cdot \sigma\left(\mathbf{W}_1 \mathbf{x}_i\right)$$

Objective: find $\mathbf{W}$ for which $\tilde{\mathbf{y}}_i(\mathbf{x}_i)$ is close to $\mathbf{y}_i$ for all $i \leq N$.

Deep learning is awesome ✓

- Simple and modular
- Huge attention from practitioners and engineers
- Great software tools
- Scales with data and compute
- Real-world impact

... but has many issues ✗

- What does a model not know?
- Uninterpretable black-boxes
- Easily fooled (AI safety)
- Crucially relies on big data

Surprisingly tying all these together: no uncertainty!

Uncertainty

UNIVERSITY OF OXFORD

- Why should I care about uncertainty?
  - We train a model to recognise dog breeds

Uncertainty

UNIVERSITY OF OXFORD

- Why should I care about uncertainty?
  - We train a model to recognise dog breeds
  - And are given a cat to classify

Uncertainty

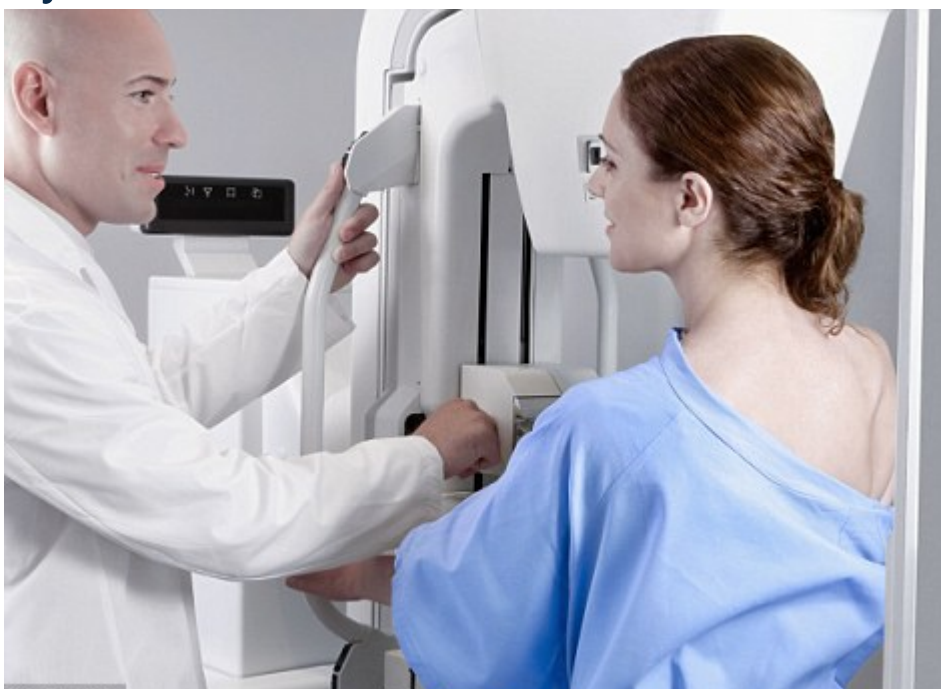
UNIVERSITY OF OXFORD

- Why should I care about uncertainty?
  - We train a model to recognise dog breeds
  - And are given a cat to classify
  - What would you want your model to do?

Uncertainty

UNIVERSITY OF OXFORD

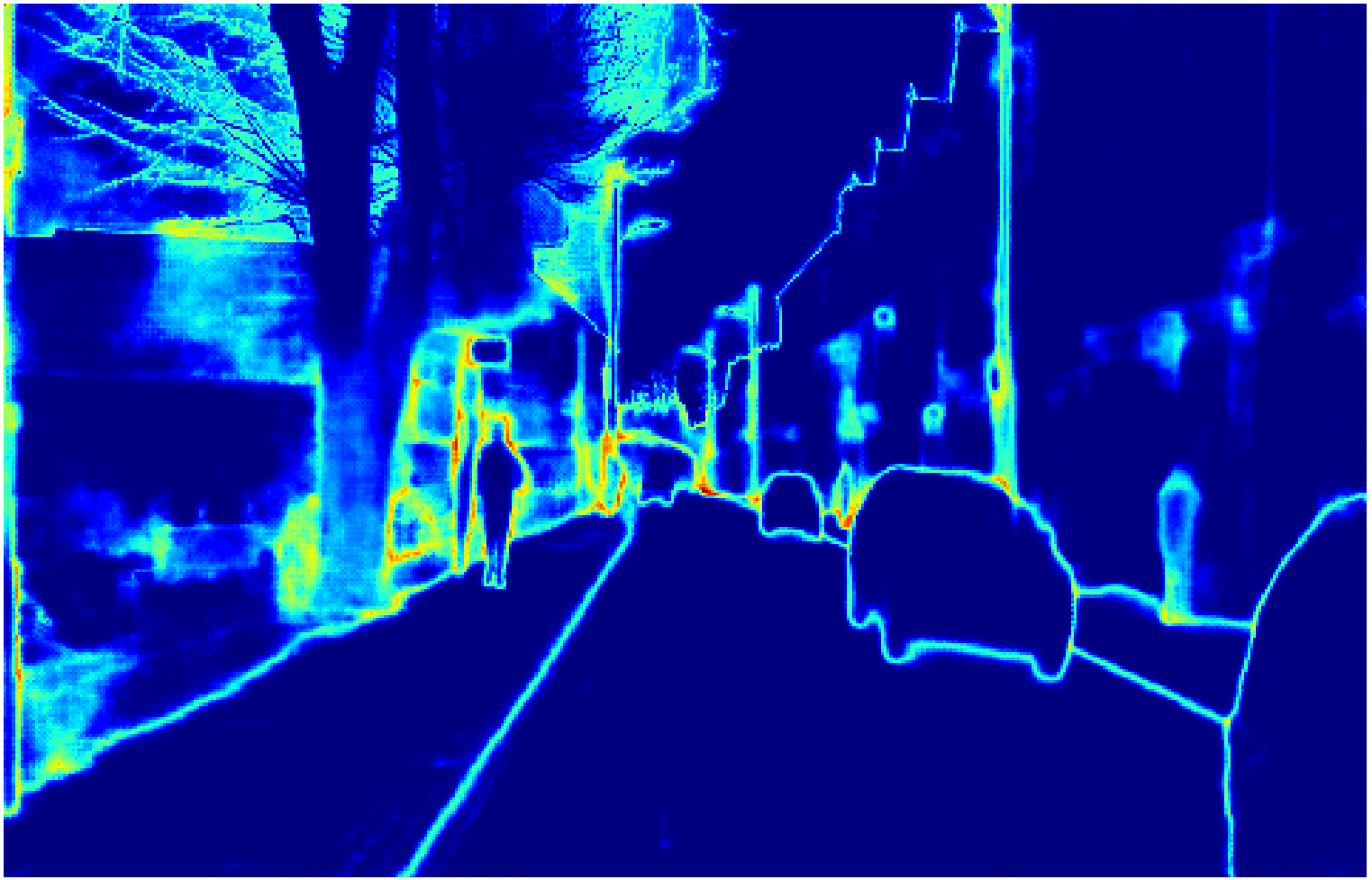
- Why should I care about uncertainty?
  - We train a model to recognise dog breeds
  - And are given a cat to classify
  - What would you want your model to do?
  - Similar problems in *decision making*, *physics*, *life science*, etc. We need a way to tell **what our model knows** and what not.



Uncertainty

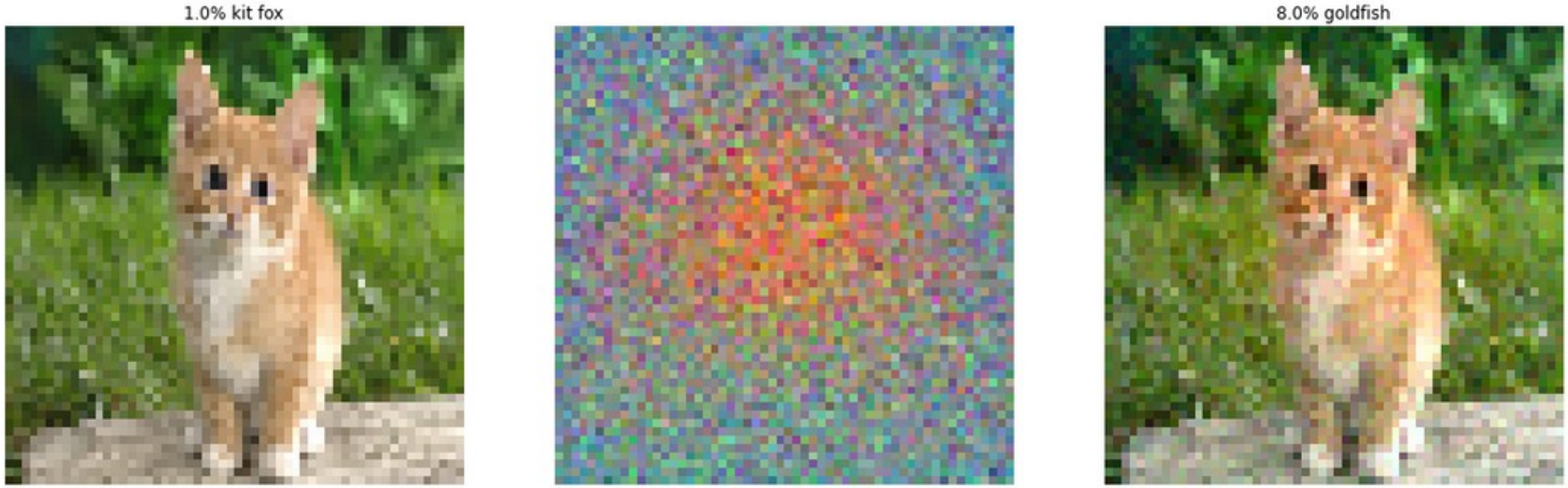
UNIVERSITY OF OXFORD

- Why should I care about uncertainty?
- Uncertainty gives insights into the black-box when it fails —where am I not certain?



Uncertainty

- Why should I care about uncertainty?
- Uncertainty gives insights into the black-box when it fails —where am I not certain?
- Uncertainty is even be useful to identify when attacked with adversarial examples!



- Lastly, need less data if label only where **model is uncertain**: wear-and-tear in robotics, expert time in medical analysis

Uncertainty

- Why should I care about uncertainty?
- Uncertainty gives insights into the black-box when it fails —where am I not certain?
- Uncertainty is even be useful to identify when attacked with adversarial examples!
- Lastly, need less data if label only where **model is uncertain**: wear-and-tear in robotics, expert time in medical analysis

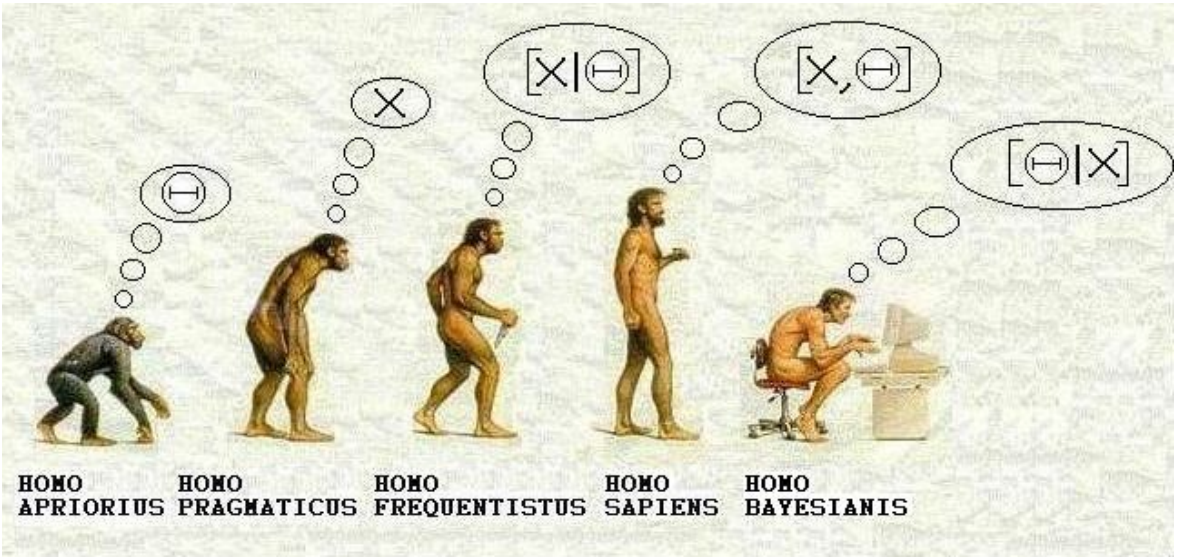
Pillar II: Bayes

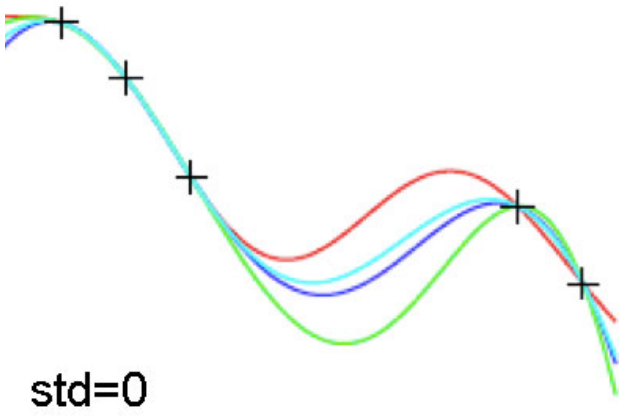
The language of uncertainty

- Probability theory
- Specifically *Bayesian probability theory* (1750!)

When applied to *Information Engineering*...

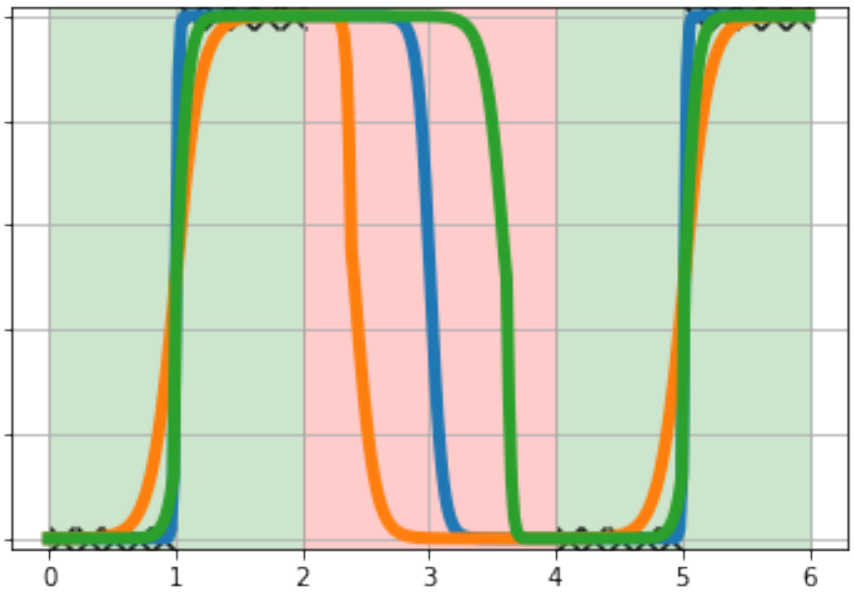
- Bayesian modelling





- Built on solid mathematical foundations
- Orthogonal to deep learning...

A simple way to tie the two pillars together



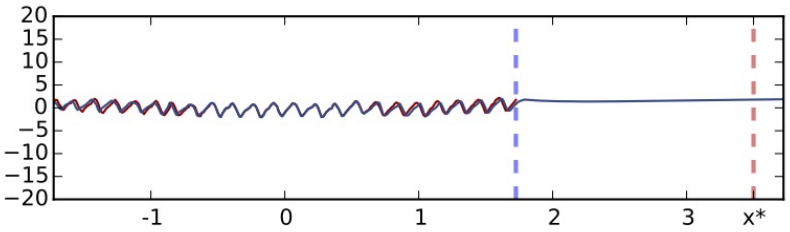
- ensemble / do dropout **at test time**
- **repeat 10 times**
- and look at **variance/entropy** (disagreement) of the sampled outputs at text location x^*.

Or in Python:

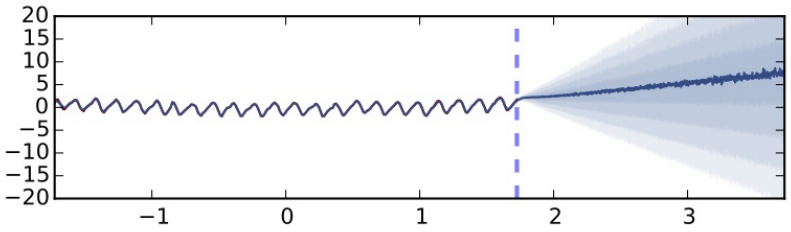
```
1 y = []
2 for _ in xrange(10):
3     y.append(model.output(x_test, dropout=True))
4 y_mean = numpy.mean(y)
5 y_var = numpy.var(y)
```

What would be the CO_2 concentration level in Mauna Loa, Hawaii, *in 20 years' time*?

Normal deep learning:



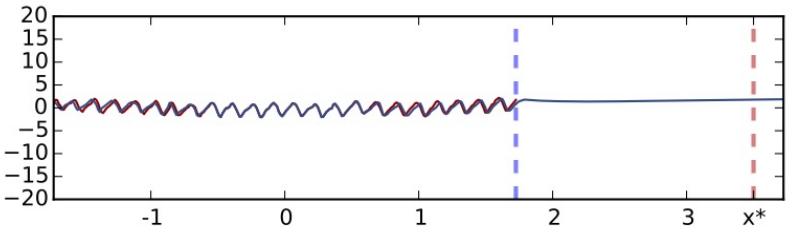
Bayesian perspective:



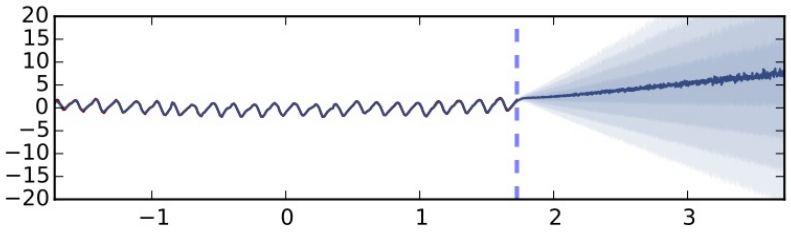
Uncertainty in deep learning

What would be the CO_2 concentration level in Mauna Loa, Hawaii, *in 20 years' time*?

Normal deep learning:



Bayesian perspective:



Now that we can capture uncertainty in deep learning, what can we do with this?

- Refuse to answer questions when we don't know the answer with generative QA
- Detect ambiguous questions, and ask for clarifications
- Highlight sentences which might be incorrect in long generations

UNIVERSITY OF OXFORD

Wait, not so fast! with LLM foundation models,

- we can't *afford* to train multiple models for an ensemble
- training even a single model can cost millions of $
- worse, the entropy for multiple utterances in natural language generation doesn't actually capture what we think it does...
  - question $x =$ "What is the capital of France?"
    answer s:

Answer $\mathbf{s}$	Likelihood $p(\mathbf{s} \mid x)$
Paris	0.5
Rome	0.4
London	0.1

Answer $\mathbf{s}$	Likelihood $p(\mathbf{s} \mid x)$
Paris	0.5
It's Paris	0.4
London	0.1

Which model is more uncertain about the answer?

UNIVERSITY OF OXFORD

Wait, not so fast! with LLM foundation models,

- we can't *afford* to train multiple models for an ensemble
- training even a single model can cost millions of $
- worse, the entropy for multiple utterances in natural language generation doesn't actually capture what we think it does...
  - question $x =$ "What is the capital of France?"
    answer s:

Answer $\mathbf{s}$	Likelihood $p(\mathbf{s} \mid x)$
Paris	0.5
Rome	0.4
London	0.1
Entropy	0.31

Answer $\mathbf{s}$	Likelihood $p(\mathbf{s} \mid x)$
Paris	0.5
It's Paris	0.4
London	0.1
Entropy	0.31

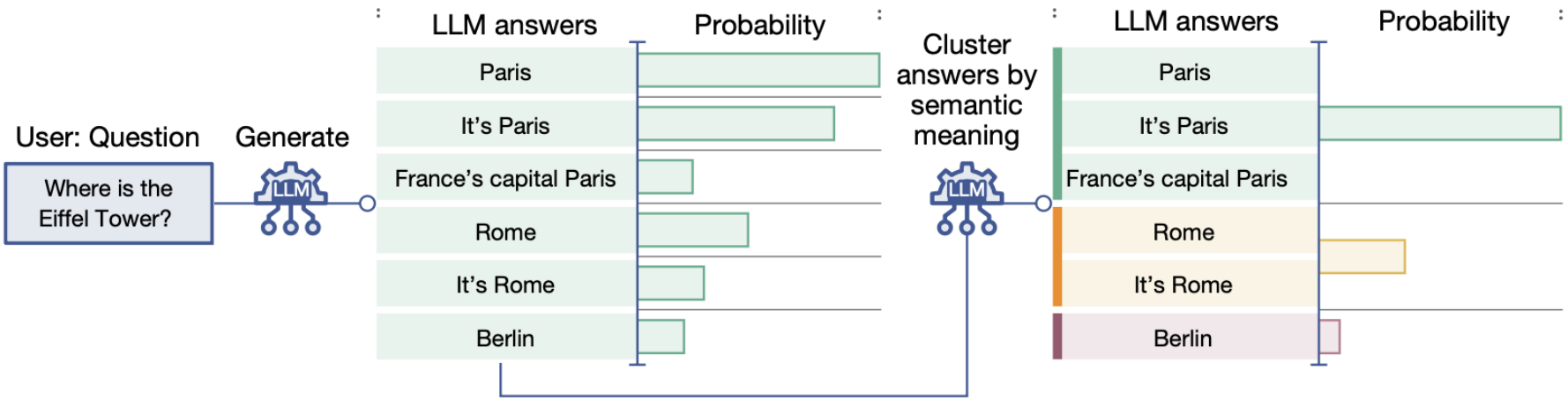
Left model has high prob for multiple cities,
Right model is very certain the answer is Paris,
Yet both models have same entropy for output!

Uncertainty in foundation models

What's going on?

- Entropy of output utterances is high when we have *syntactic diversity*, whereas we care only about *semantic diversity*.

Instead, let's define equivalence classes *c* over the outcome space (output utterances *s*) and match together utterances which are semantically equivalent.



Uncertainty in foundation models

What's going on?

- Entropy of output utterances is high when we have *syntactic diversity*, whereas we care only about *semantic diversity*.

Instead, let's define equivalence classes c over the outcome space (output utterances s) and match together utterances which are semantically equivalent.

We change our model's output to be the semantic class C, marginalising over utterances with same meaning

$$p(C = c|x) = \sum_{s \in c} p(s|x)$$

Entropy of modified model:

$$\mathrm{SE}(x) = -\sum_{c} P(c|x) \log P(c|x) = -\sum_{c} \left[\sum_{s \in c} p(s|x) \right] \log \left[\sum_{s \in c} p(s|x) \right]$$

Uncertainty in foundation models

We need to evaluate:

$$\mathrm{SE}(x) = -\sum_{c} P(c|x) \log P(c|x).$$

Sum is intractable... Let's approx w/ Rao–Blackwellized Monte Carlo integration over the semantic equivalence classes c:

$$\mathrm{SE}(x) \approx -\sum_{i=1}^{|C|} \hat{P}(c_i|x) \log \hat{P}(c_i|x)$$

with $|C|$ clusters c_i from N generated answers $s_n \sim S|x$, $\hat{P}(c_i|x) = \frac{P(c_i|x)}{\sum_c P(c|x)}$ and $P(c_i|x) = \sum_{s_n \in c_i} P(s_n|x)$.

Uncertainty in foundation models

In practice we implement the algorithm as follows:

1. Generate a set of answers from the model $\{s_n\}_n$
2. Cluster by semantic equivalence $\{C_i\}_i$ and compute $\hat{P}(c_i|x)$
3. Compute entropy w.r.t. *equivalence classes* C_i

$$\mathrm{SE}(x) \approx -\sum_{i=1}^{|C|} \hat{P}(c_i|x) \log \hat{P}(c_i|x)$$

Now the right model has lower uncertainty than the left model, as we'd expect:

(a) Scenario 1: No semantic equivalence

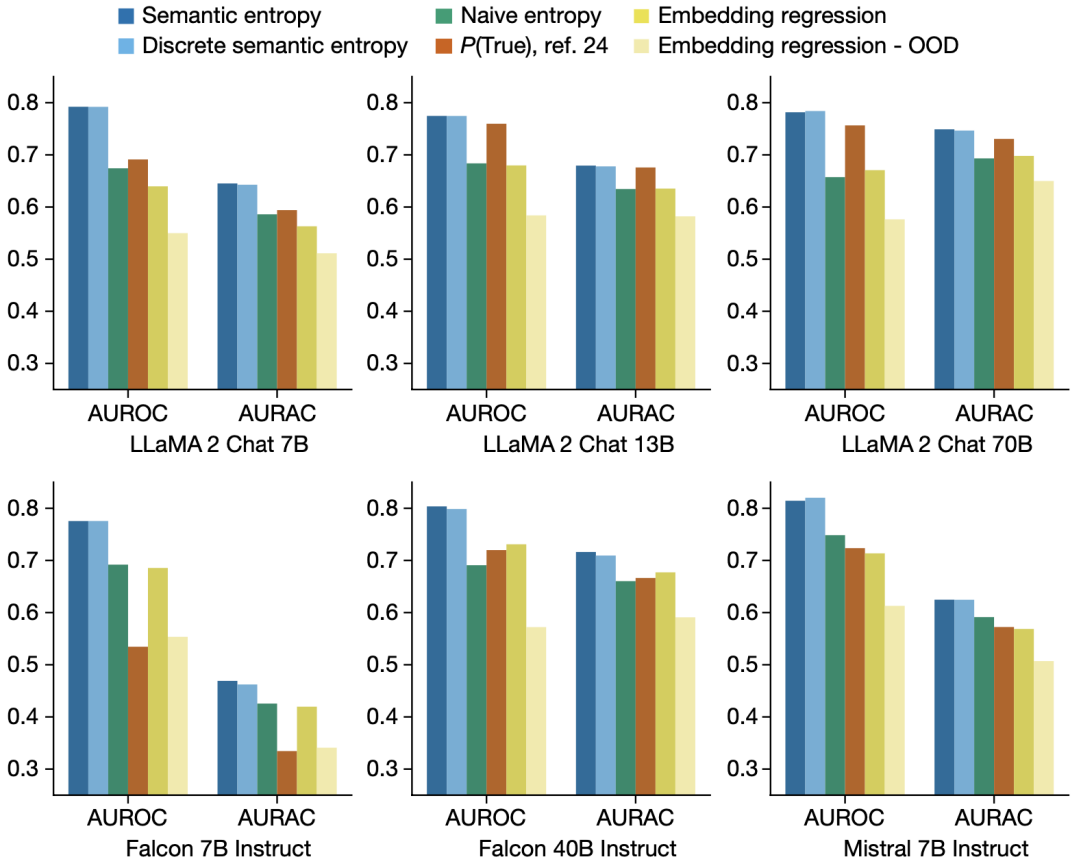
Answer $\mathbf{s}$	Likelihood $p(\mathbf{s} \mid x)$	Semantic likelihood $\sum_{\mathbf{s}\in c} p(\mathbf{s} \mid x)$
Paris	0.5	0.5
Rome	0.4	0.4
London	0.1	0.1
Entropy	0.31	0.31

(b) Scenario 2: Some semantic equivalence

Answer $\mathbf{s}$	Likelihood $p(\mathbf{s} \mid x)$	Semantic likelihood $\sum_{\mathbf{s}\in c} p(\mathbf{s} \mid x)$
Paris	0.5	0.9
It's Paris	0.4	
London	0.1	0.1
Entropy	0.31	0.16

Selective question answering

With *practical* uncertainty at hand, we can **refuse to answer questions when we don't know the answer**:

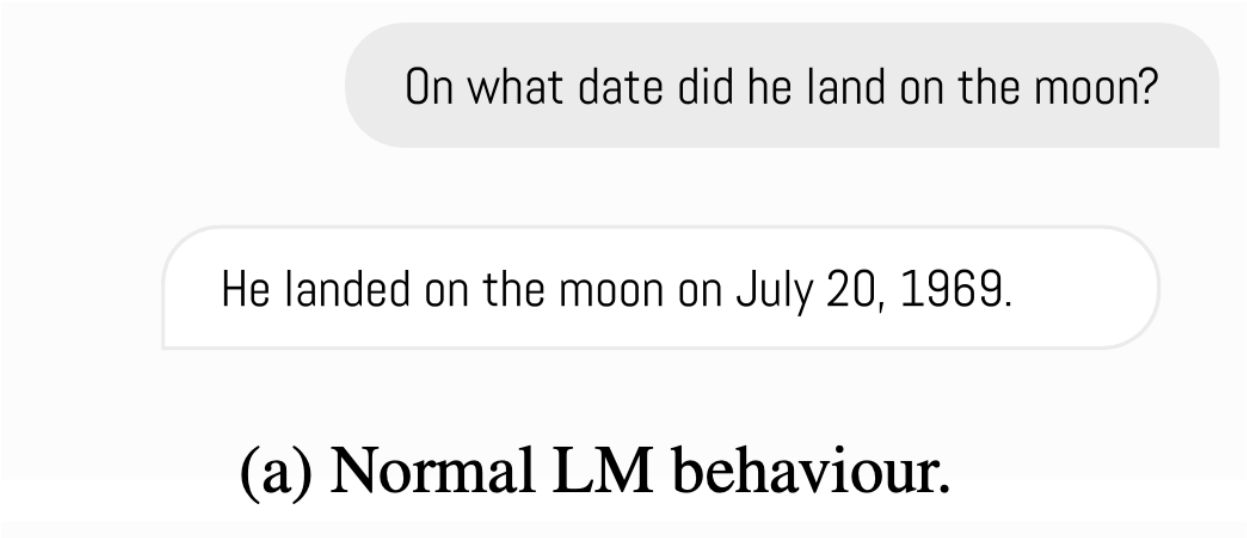


Area under curve (AUC) of binary event "answer is correct" at different uncertainty thresholds (ROC), and performance improvement of a system that refuses to answer questions which are judged "likely confabulations" (RAC). Higher is better. Baselines: **naive entropy** computes token-wise entropy; **P(True)** looks at prob of next token being 'True' when few-shot prompted to compare a main answer with 'brainstormed' alternatives; **Emg regression** is a supervised baseline where logistic regression predicts if the model answered correctly.

Nature 2024 [Farquhar, Kossen, Kuhn, & Gal]

Using uncertainty in FMs

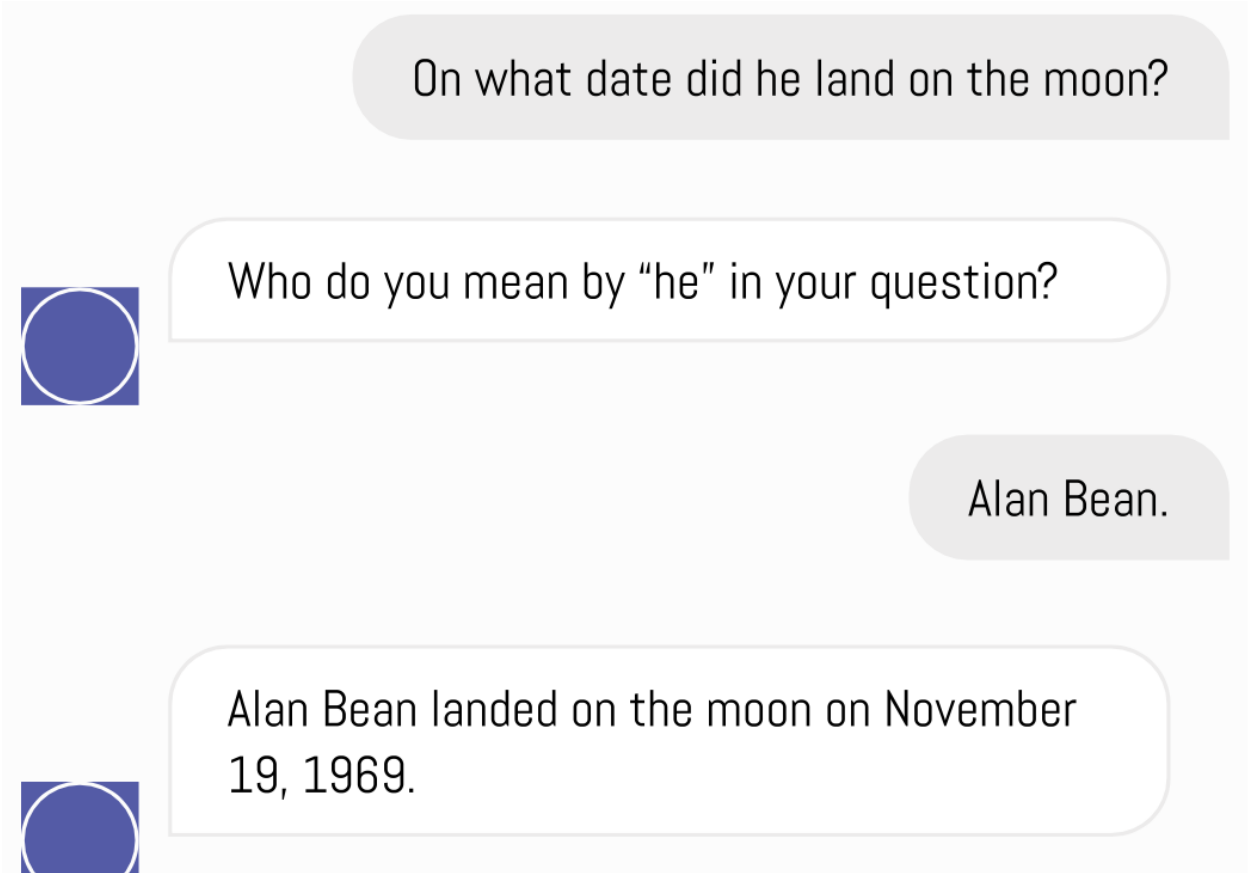
We have a new tool to detect when the model doesn't know the answer to a question. How can we use such a tool?



(a) Normal LM behaviour.

Using uncertainty in FMs

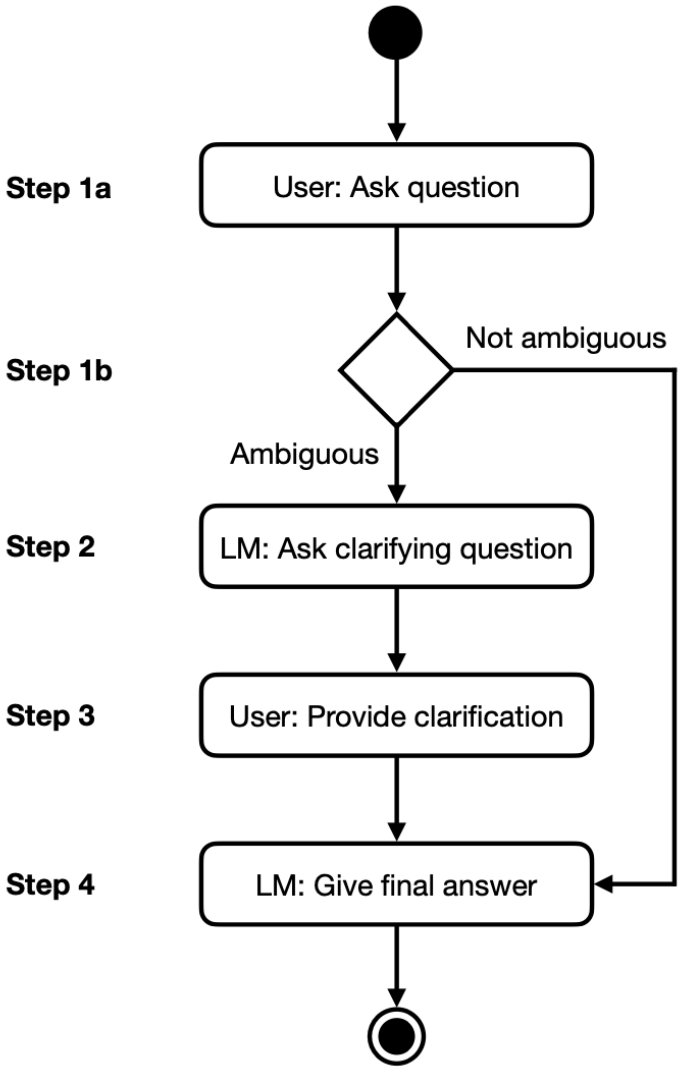
We have a new tool to detect when the model doesn't know the answer to a question. How can we use such a tool?



(b) Desired behaviour

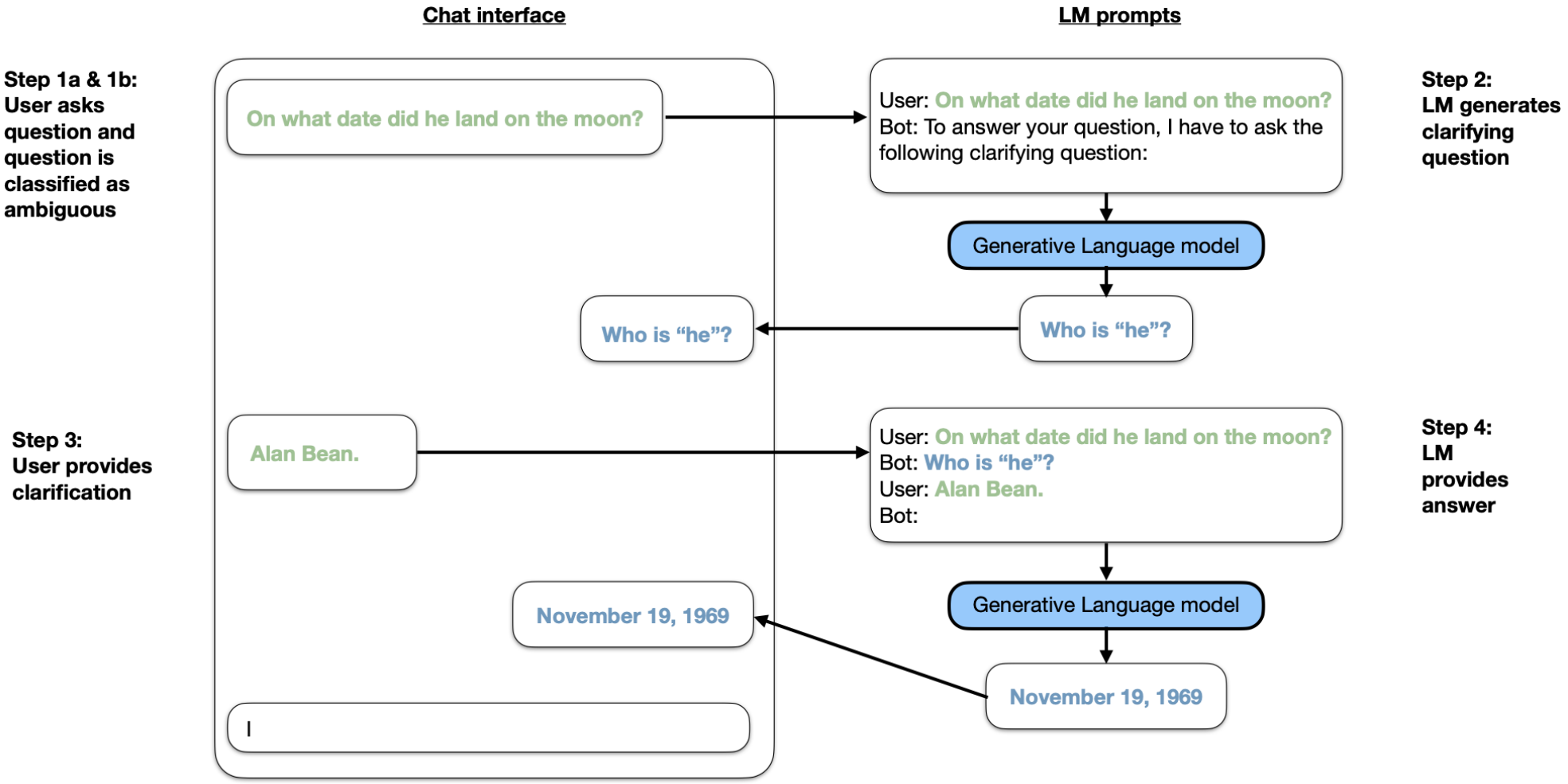
Using uncertainty in FMs

We have a new tool to detect when the model doesn't know the answer to a question. How can we use such a tool?



Clarifying questions

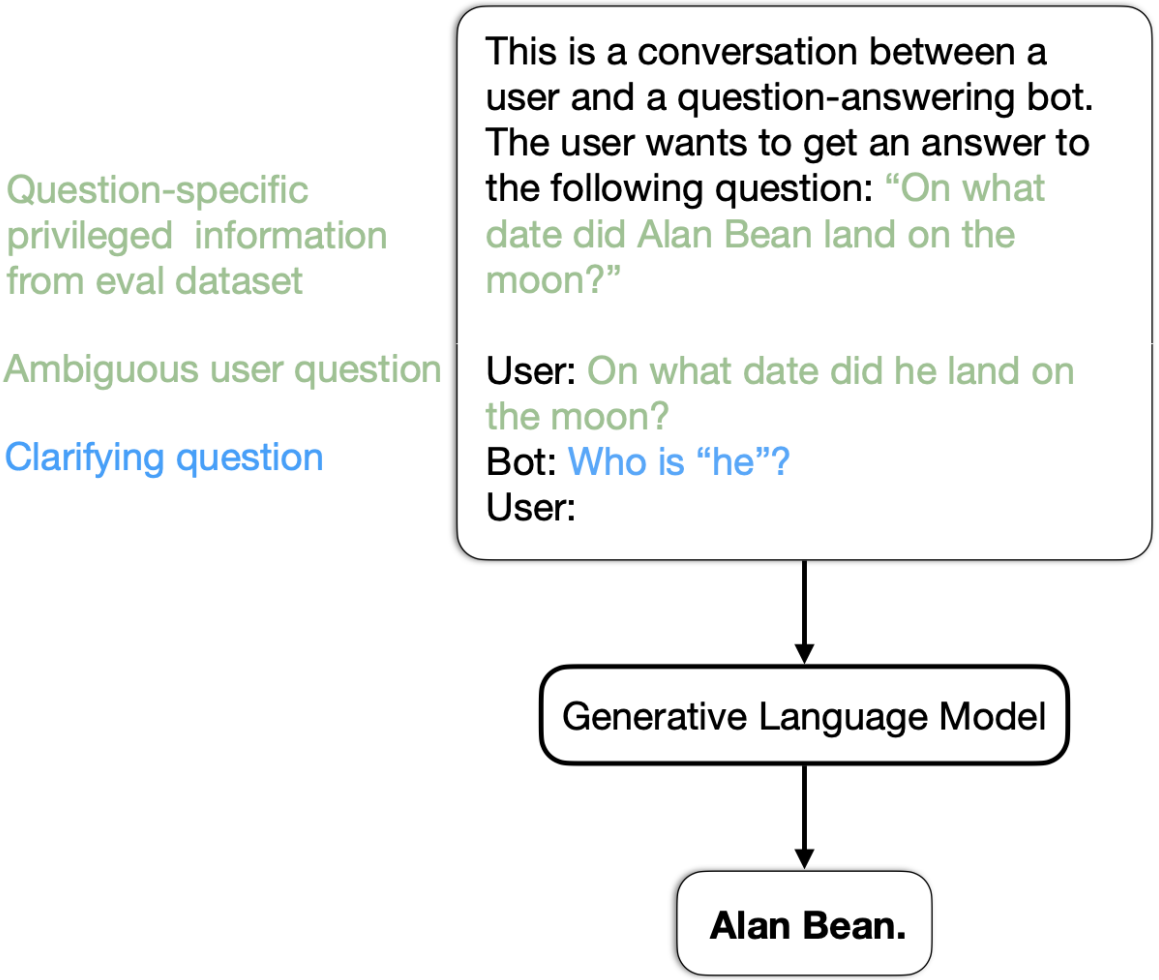
We can generate clarifying qs when we detect a question to be ambiguous:



Clarifying questions

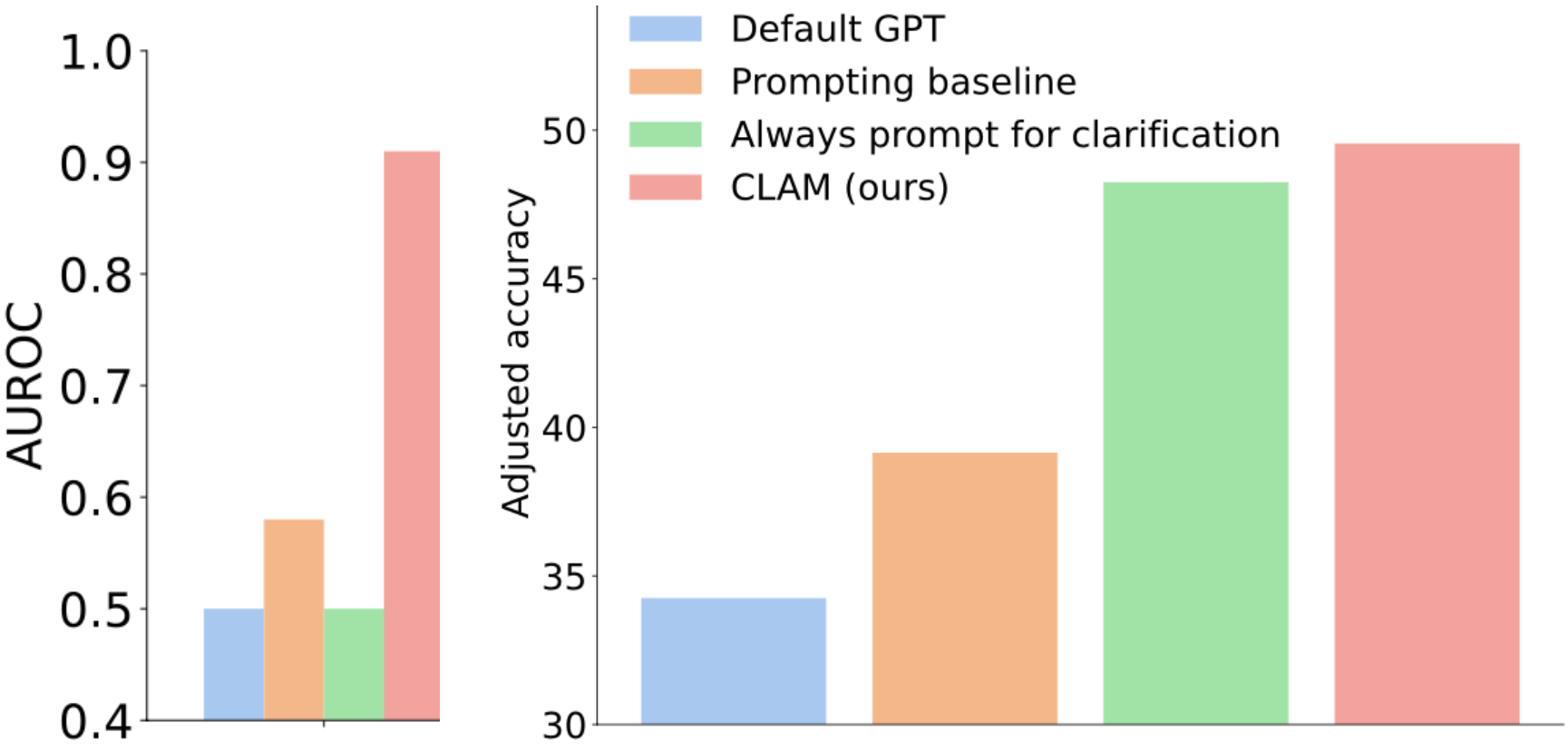
How to *eval*? (can't generate user-responses for each q in corpus)

- Borrow ideas from active learning – use *oracle* that has access to *privileged information*.



Clarifying questions

We can avoid asking for clarification with unambiguous qs, and for amb qs asking for clarification improves performance:



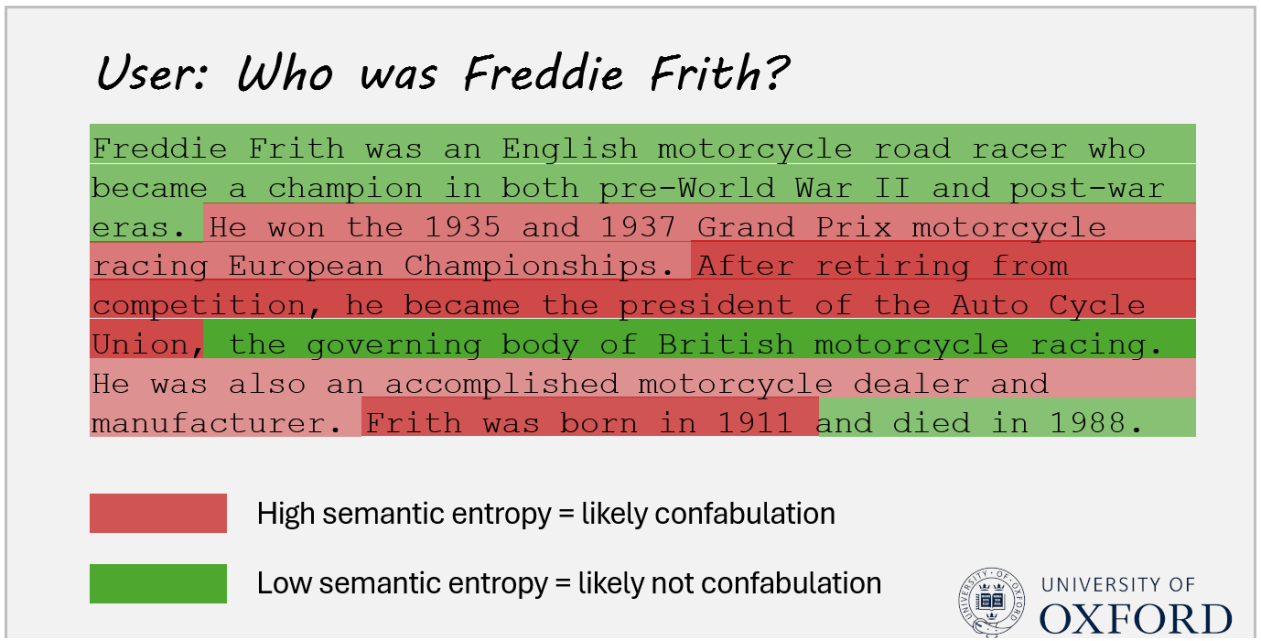
Ambiguity detection AUC and QA accuracy on dataset derived from TriviaQA (consists of ambiguous and unambiguous questions)

Preprint [Kuhn, Gal, & Farquhar]

Long form generation

So far we talked about detecting hallucinations in short-form generation. But what if we need to generate **longer texts**?

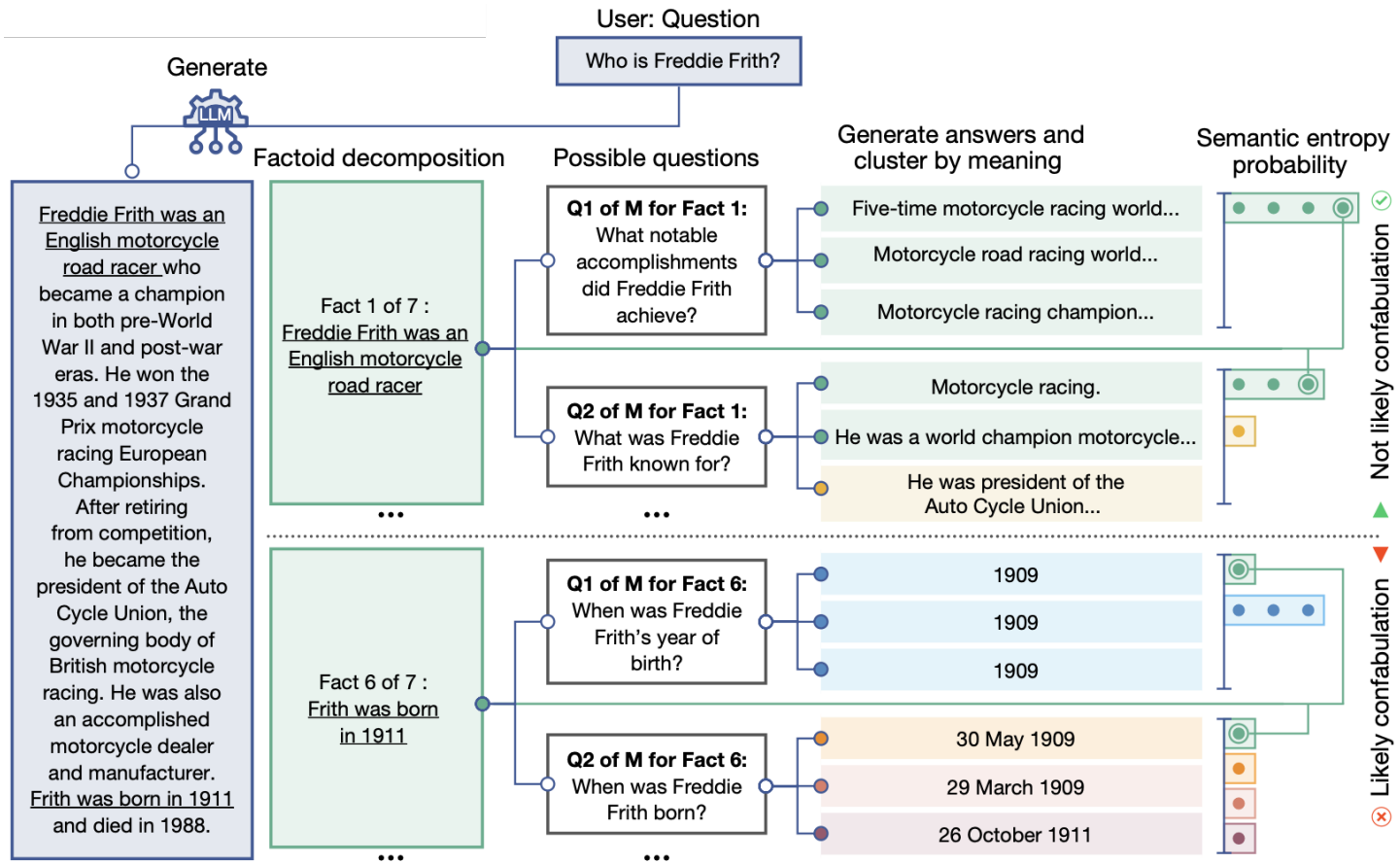
- Surprisingly challenging.. generating the long text multiple times incorrectly assigns high uncertainty because **text structure changes** (eg, in a biography, different facts might be mentioned)
- But with the the tools above, we can reduce long text into short-form uncertainty estimation



Long form generation

UNIVERSITY OF OXFORD

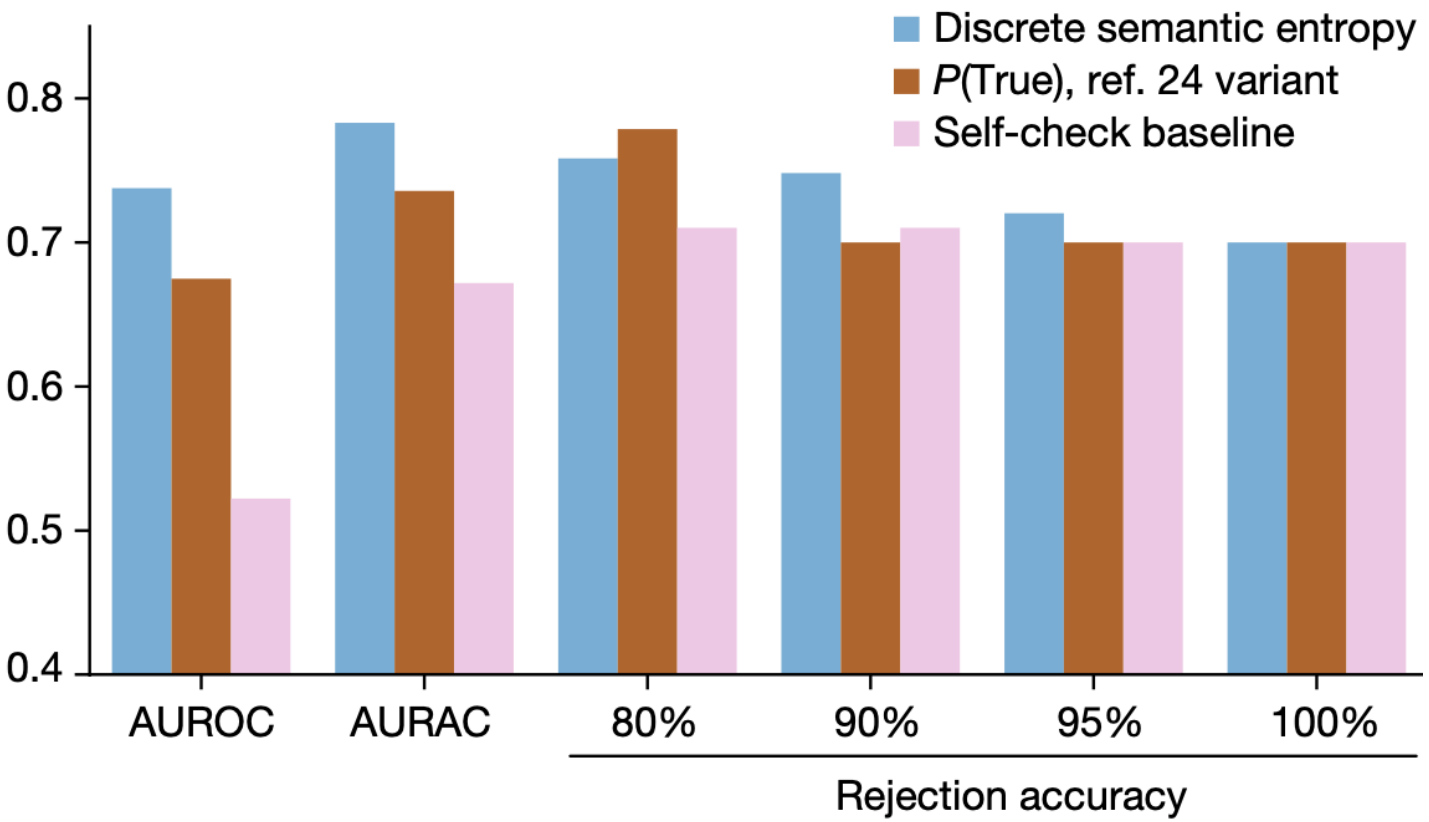
We reduce long form uncertainty estimation to short form:



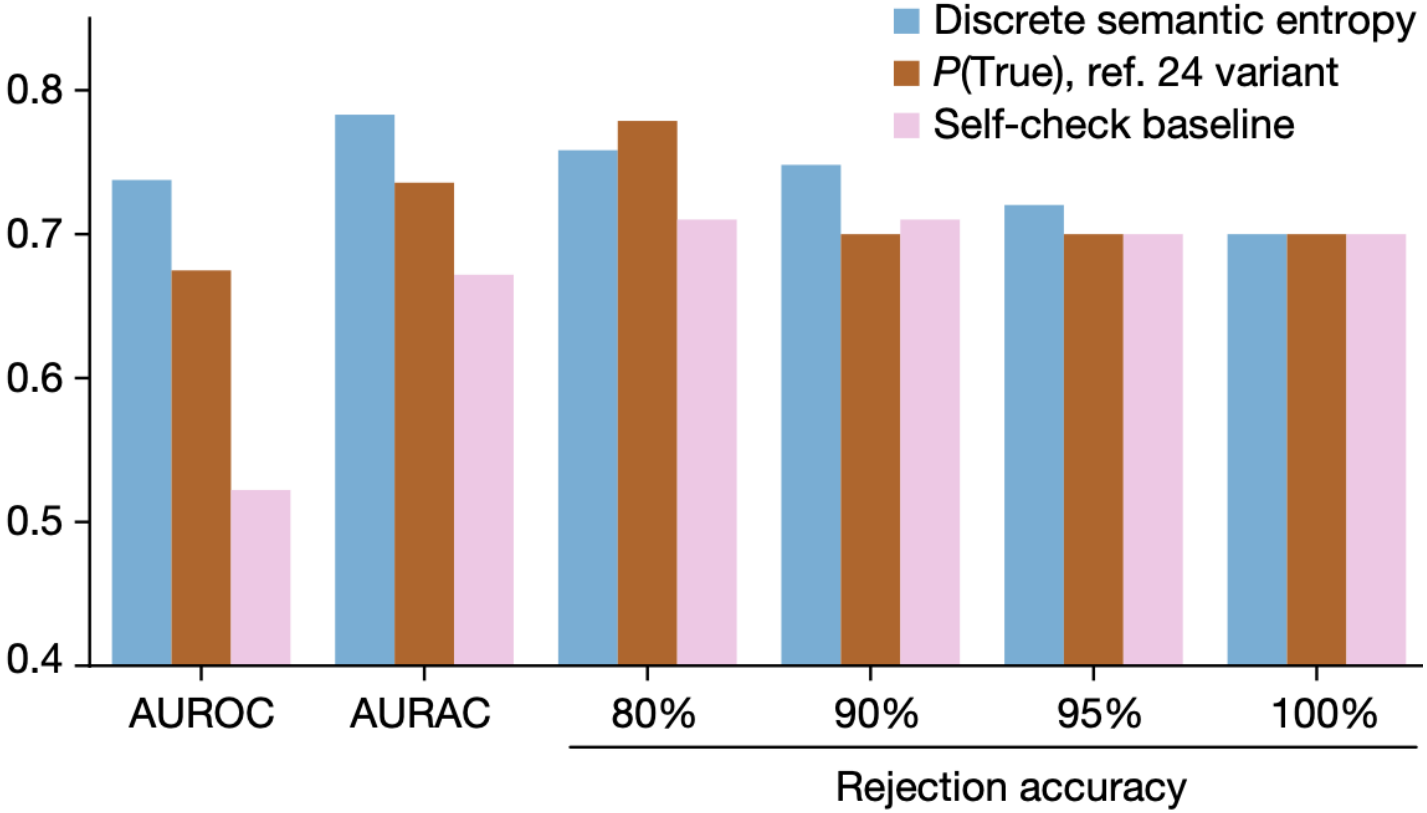
Here we generate multiple question to target a different aspect of each fact.

GPT-4 confabulations in biographies:

- We developed a dataset of **biographies** generated from GPT-4
- From each we **automatically extract factual claims** which we manually label as true or false
- We then run each confabulation detection method on each biography and look at AUROC, AURAC, and 'correct answer' accuracy at % rejected (100% = all qs being answered)

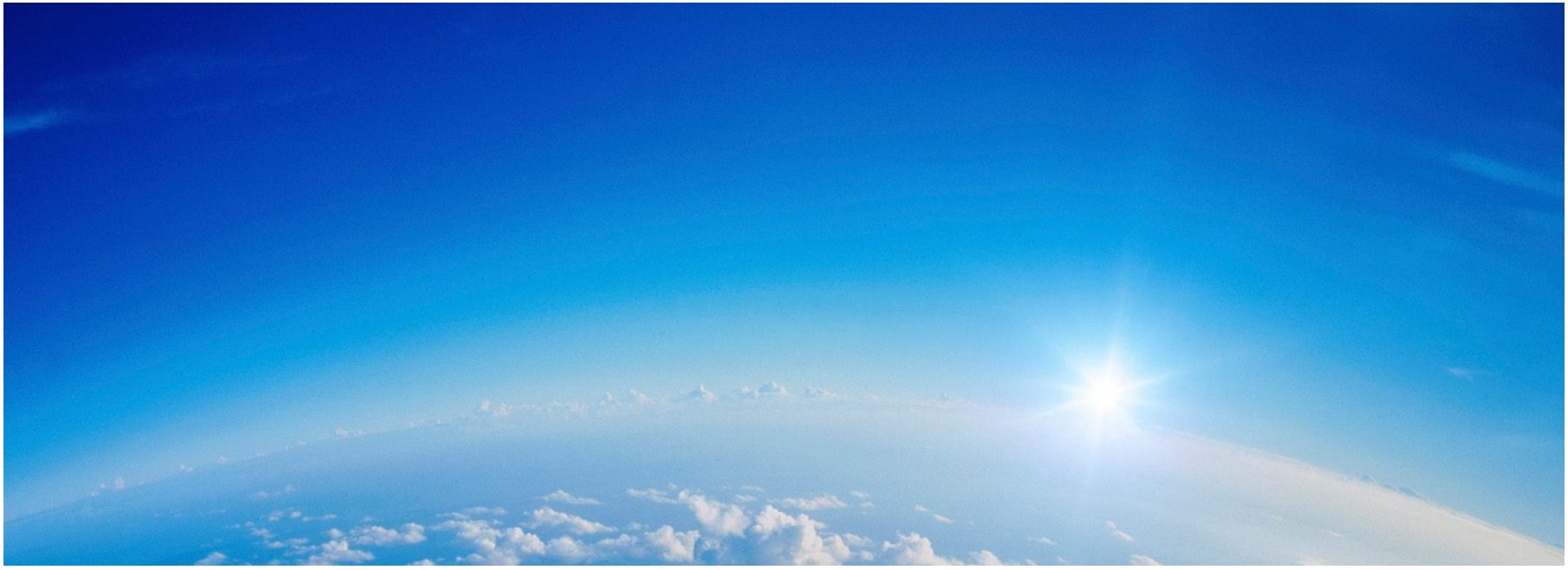


GPT-4 confabulations in biographies:



- Both P(True) and self-check use our factoid decomposition introduced above
- SE captures confabulations alone, while P(True) captures some confabulations but also some systematic errors (cf 80%)
- Self-check asks LLM if factoid is true; this ought to perform very well if we believe “LLMs mostly know what they don’t know”!

More Research



- **Develop new tools** and help others build **safe and robust ML** for responsible deployment in industry and academia
- **Autonomous driving** (OOD scenes), **medical applications** (BRaTS), **safe RL**, ...

and much, much, more.