


SEMANTIC UNCERTAINTY: LINGUISTIC INVARIANCES FOR UNCERTAINTY ESTIMATION IN NATURAL LANGUAGE GENERATION

Lorenz Kuhn, Yarin Gal, Sebastian Farquhar

OATML Group, Department of Computer Science, University of Oxford

`lorenz.kuhn@cs.ox.ac.uk` 

ICLR'23

8886, top25%

Citation 212

Problem

- Measuring uncertainty in natural language is challenging.
 - Most machine learning problems have mutually **exclusive outputs**. An **image** in class 17 is not class 29 as well; a **regression** output of 23.1 is not anything else; an **RL agent** going left does not go right.
 - In contrast, for free-form text generation an output usually **means the same thing as many other outputs**. For example, “The capital of France is Paris” means the same thing as “France’s capital is Paris”.
 - The key challenges come from the importance in language of **meanings** and **form**. This corresponds to what linguists and philosophers call the *semantic content* of a sentence and its **syntactic** or **lexical** form. **Foundation models output token-likelihoods—representing lexical confidence. But for almost all applications we care about meanings!**

Sentence A	Sentence B	Equivalence		
		Lexical	Syntactic	Semantic
Paris is the capital of France.	Paris is the capital of France.	✓	✓	✓
	Berlin is the capital of France.		✓	
	France’s capital is Paris.			✓

Semantic Likelihoods

- **Semantic likelihoods** vs. sequence-likelihoods
 - Probabilities attached to **meanings of text**.
 - Introduce an algorithm for clustering sequences that mean the same thing based on the principle that two sentences mean the same thing **if you can infer each from the other**.
 - Uncertainty over different meanings

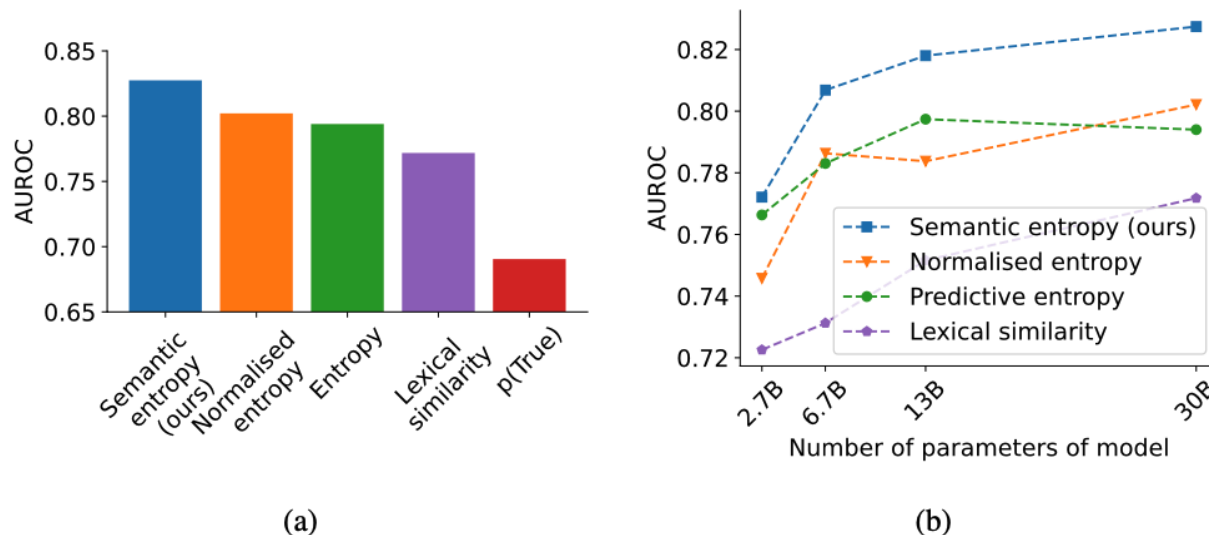


Figure 1: (a) Our semantic entropy (blue) predicts model accuracy better than baselines on the free-form question answering data set TriviaQA (30B parameter OPT model). Normalised entropy reimplements single-model variant of [Malinin & Gales \(2020\)](#), lexical similarity measures the average Rouge-L in a sampled set of answers for a given question analogously to [Fomicheva et al. \(2020\)](#), entropy and $p(\text{True})$ reimplement [Kadavath et al. \(2022\)](#). (b) Our method's outperformance increases with model size while also doing well for smaller models.

Active Prompt

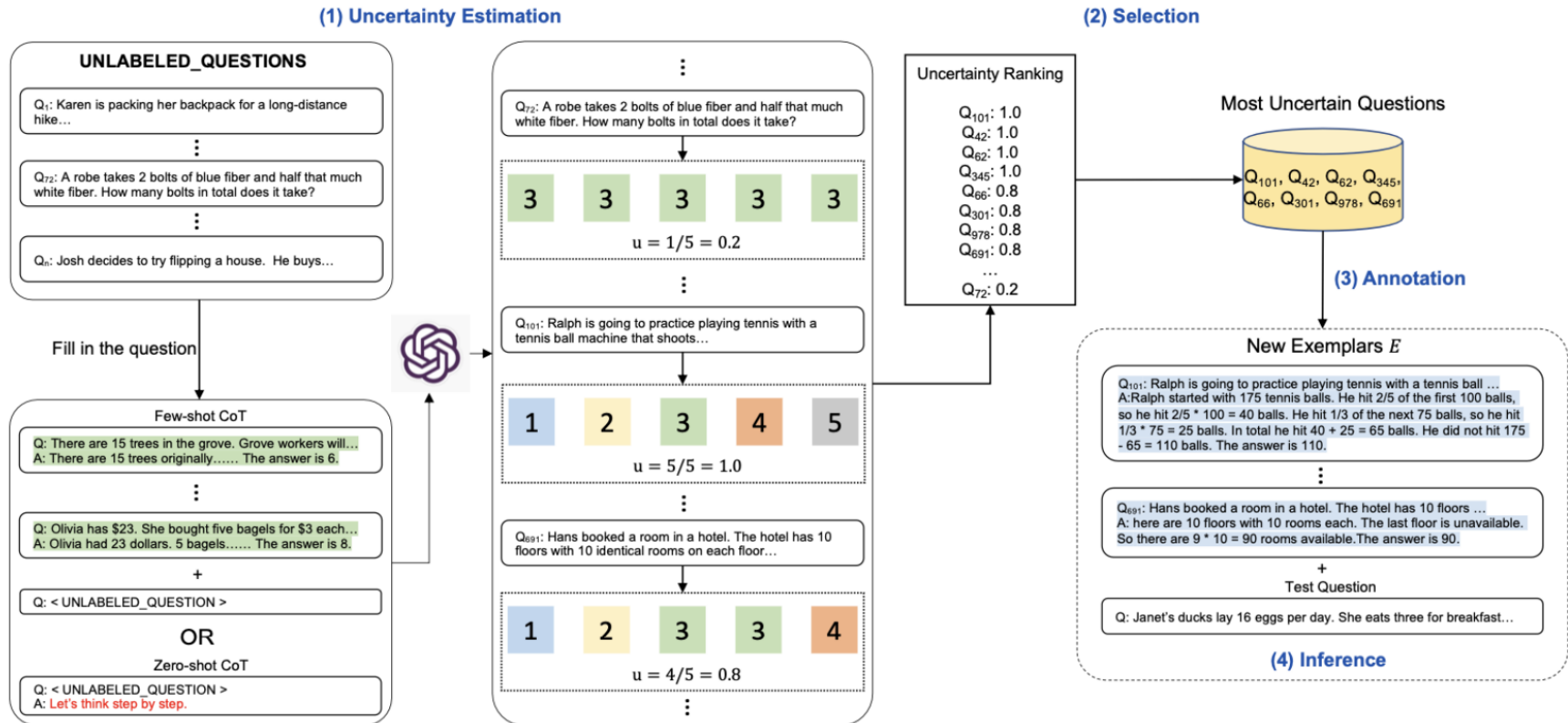


Figure 1: Illustrations of our proposed approach. There are four stages. **(1) Uncertainty Estimation:** with or without a few human-written chain-of-thoughts, we query the large language model k ($k = 5$ in this illustration) times to generate possible answers with intermediate steps for a set of training questions. Then we calculate the uncertainty u based on the k answers via an uncertainty metric (we use disagreement in this illustration). **(2) Selection:** according to the uncertainty, we select the most uncertain questions for annotation. **(3) Annotation:** we involve humans to annotate the selected questions. **(4) Inference:** infer each question with the new annotated exemplars.

Semantic Uncertainty

1. **Generation:** Sample M sequences $\{s^{(1)}, \dots, s^{(M)}\}$ from the predictive distribution of a large language model given a context x .
2. **Clustering:** Cluster the sequences which mean the same thing using our bi-directional entailment algorithm.
3. **Entropy estimation:** Approximate semantic entropy by summing probabilities that share a meaning following Eq. (2) and compute resulting entropy. This is illustrated in Table 1.

In general, any natural language inference classification system (NLI) can be used for our bidirectional entailment clustering algorithm. In our case, we use a Deberta-large model (He et al., 2020a) that is fine-tuned on the NLI data set MNLI (Williams et al., 2017). For each pair of sequences in our set of samples, s and s' , we detect whether it is possible to infer the concatenation of the context and s from the concatenation of the context and s' and vice versa. To do this we concatenate each of the two question/answer pairs, and then concatenate them both together separated by a special token. The Deberta model then classifies this sequence into one of: entailment, neutral, contradiction. We compute both directions, and the algorithm returns equivalent if and only if both directions were entailment. Algorithm pseudocode is provided in Appendix A.2.

Because this component is novel, we confirm in Appendix B.2 that the bidirectional entailment classifier works by manually labelling 300 generations for semantic equivalence, finding an accuracy of 92.7% on TriviaQA and 95.5% on CoQA.

Semantic Uncertainty

Algorithm 1 Bidirectional Entailment Clustering

Require: context x , set of seqs. $\{s^{(2)}, \dots, s^{(M)}\}$, NLI classifier \mathcal{M} , set of meanings $C = \{\{s^{(1)}\}\}$

```
for  $2 \leq m \leq M$  do
  for  $c \in C$  do
     $s^{(c)} \leftarrow c_0$ 
     $\text{left} \leftarrow \mathcal{M}(\text{cat}(x, s^{(c)}, "<g/>", x, s^{(m)}))$ 
     $\text{right} \leftarrow \mathcal{M}(\text{cat}(x, s^{(m)}, "<g/>", x, s^{(c)}))$ 
    if  $\text{left}$  is entailment and  $\text{right}$  is entailment then
       $c \leftarrow c \cup s^{(m)}$ 
    end if
  end for
   $C \leftarrow C \cup \{s^{(m)}\}$ 
end for
return  $C$ 
```

▷ Compare to already-processed meanings.
▷ Use first sequence for each semantic-class.
▷ Does old sequence entail new one?
▷ Vice versa?
▷ Put into existing class.
▷ Semantically distinct, gets own class.

Step 3: Computing the semantic entropy

Having determined the clusters of generated sequences that mean the same thing, we add their likelihoods following Eq. (2) as a way of determining the likelihood of each meaning, rather than each sequence. We then compute the semantic entropy (SE) as the entropy over the meaning-distribution

$$SE(x) = - \sum_c p(c | x) \log p(c | x) = - \sum_c \left(\left(\sum_{s \in c} p(s | x) \right) \log \left[\sum_{s \in c} p(s | x) \right] \right). \quad (3)$$

Experiments

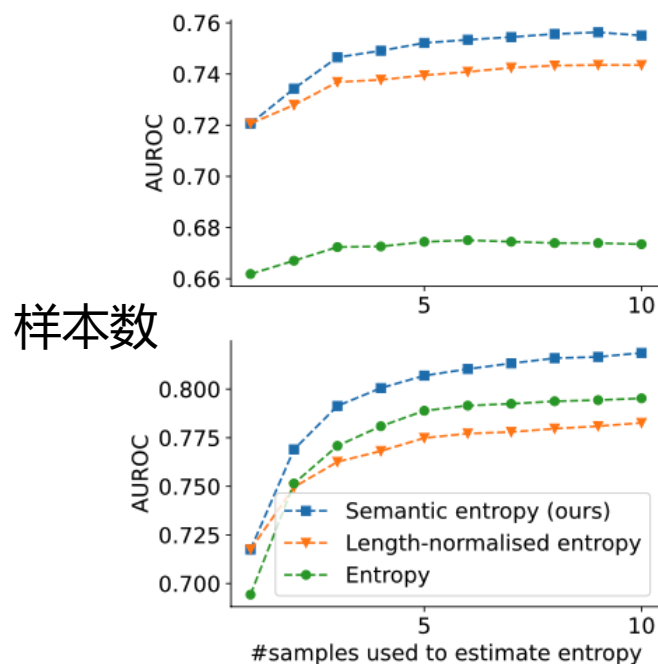
Effective uncertainty measures should offer information about how **reliable** the model's answers are—that is, very *uncertain* generations should be less likely to be *correct*.

Performance evaluation. Following prior work (e.g. Filos et al. (2019)), we evaluate uncertainty by treating uncertainty estimation as the problem of predicting whether to rely on a model generation for a given context—whether to trust an answer to a question. The area under the receiver operator characteristic curve (AUROC) metric is equivalent to the probability that a randomly chosen correct answer has a higher uncertainty score than a randomly chosen incorrect answer. Higher scores are better, with perfect uncertainty scoring 1 while a random uncertainty measure would score 0.5.

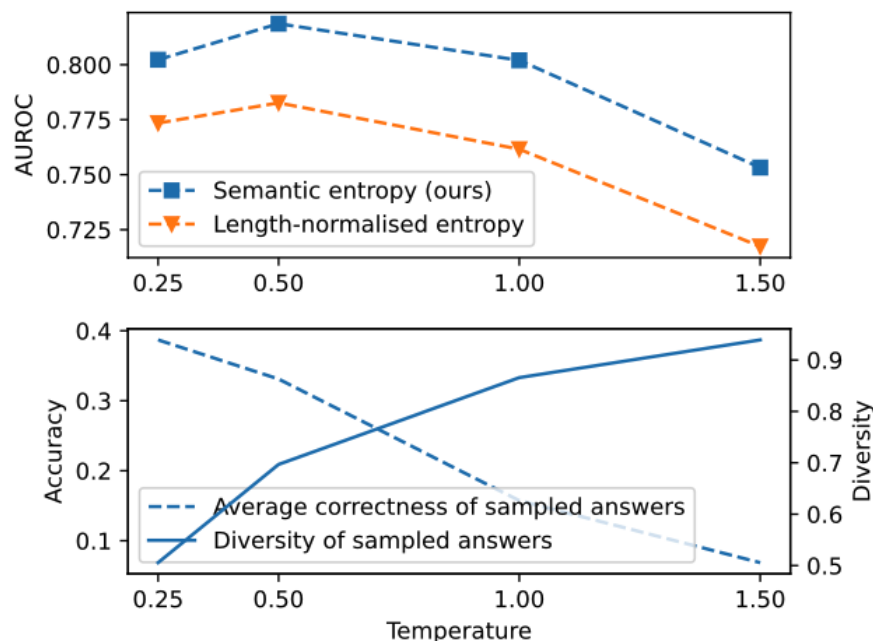
The AUROC is a better measure of uncertainty for *free-form* question answering and NLG than calibration measures like the Brier score, which are often used in classification or for multiple choice QA. This is because the language model outputs a likelihood for a given token-sequence, but not for an entire meaning. In order to estimate the Brier score, we would need to estimate the entire probability mass assigned to any possible way of saying the correct answer. This is intractable for free form text where we do not have access to probabilities about meanings. In contrast, we can estimate the entropy because it is structured as an expected information, which makes Monte Carlo integration suitable.

Table 2: Incorrectly answered questions have more semantically distinct answers than correct ones. On its own, this count is a reasonable uncertainty measure, though semantic entropy is better.

Dataset	Average # of semantically distinct answers		AUROC	
	Correctly answered	Incorrectly answered	Semantic entropy	# distinct answers
CoQA	1.27	1.77	0.77	0.66
TriviaQA	1.89	3.89	0.83	0.79



(a) (top) CoQA, (bottom) TriviaQA



(b)

Figure 3: (a) Semantic entropy makes better use of additional samples because it handles duplication better, the performance gap therefore continues to improve. (b) (bottom) Higher temperatures result in more diversity but less accurate generations. (top) The best performing uncertainty comes from an intermediate temperature that balances these two forces. Results on TriviaQA.

Detecting hallucinations in large language models using semantic entropy

<https://doi.org/10.1038/s41586-024-07421-0>

Sebastian Farquhar^{1,2}✉ Jannik Kossen^{1,2} Lorenz Kuhn^{1,2} & Yarin Gal¹

Received: 17 July 2023

Accepted: 12 April 2024

Published online: 19 June 2024

Open access

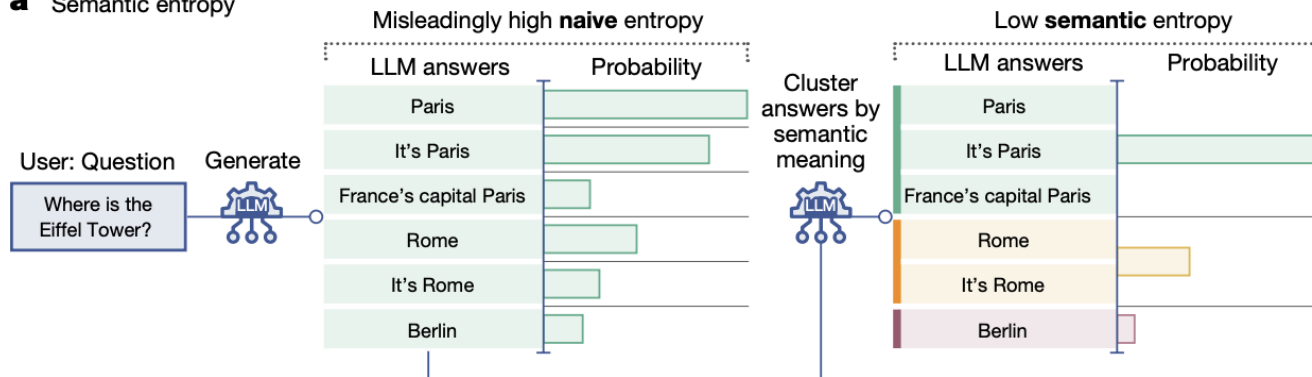
 Check for updates

Large language model (LLM) systems, such as ChatGPT¹ or Gemini², can show impressive reasoning and question-answering capabilities but often ‘hallucinate’ false outputs and unsubstantiated answers^{3,4}. Answering unreliably or without the necessary information prevents adoption in diverse fields, with problems including fabrication of legal precedents⁵ or untrue facts in news articles⁶ and even posing a risk to human life in medical domains such as radiology⁷. Encouraging truthfulness through supervision or reinforcement has been only partially successful⁸. Researchers need a general method for detecting hallucinations in LLMs that works even with new and unseen questions to which humans might not know the answer. Here we develop new methods grounded in statistics, proposing entropy-based uncertainty estimators for LLMs to detect a subset of hallucinations—confabulations—which are arbitrary and incorrect generations. Our method addresses the fact that one idea can be expressed in many ways by computing uncertainty at the level of meaning rather than specific sequences of words. Our method works across datasets and tasks without a priori knowledge of the task, requires no task-specific data and robustly generalizes to new tasks not seen before. By detecting when a prompt is likely to produce a confabulation, our method helps users understand when they must take extra care with LLMs and opens up new possibilities for using LLMs that are otherwise prevented by their unreliability.

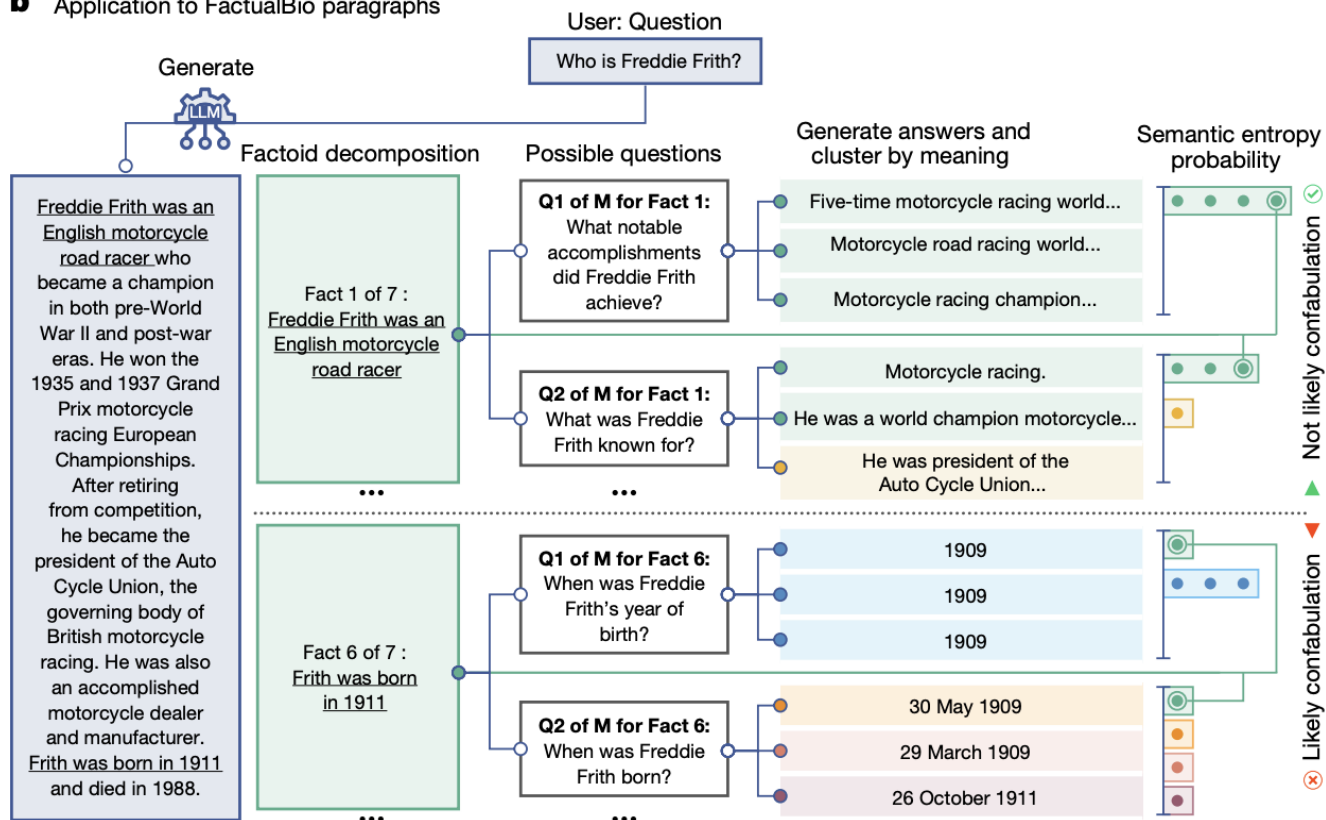
Nature’23

Pipeline

a Semantic entropy



b Application to FactualBio paragraphs



Results

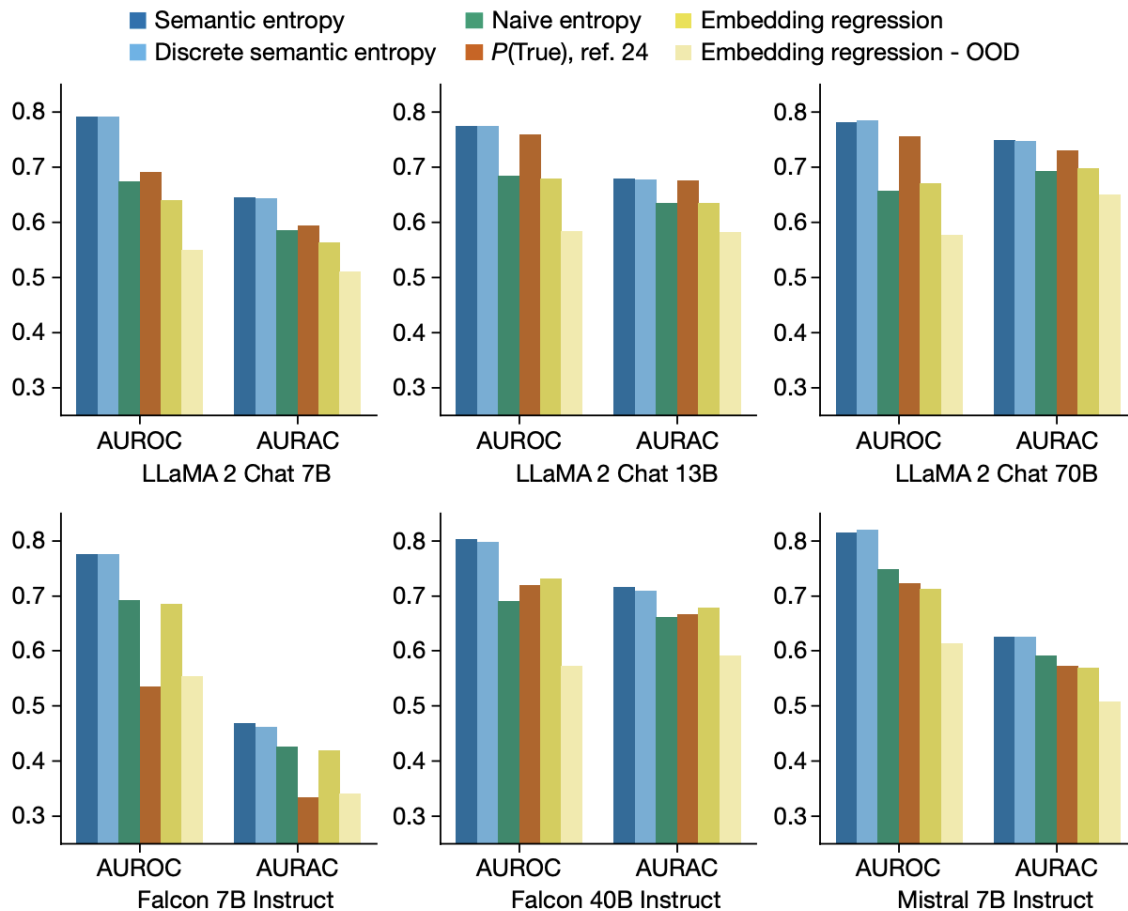


Fig. 2 | Detecting confabulations in sentence-length generations. Semantic entropy outperforms leading baselines and naive entropy. AUROC (scored on the y-axes) measures how well methods predict LLM mistakes, which correlate with confabulations. AURAC (likewise scored on the y-axes) measures the

performance improvement of a system that refuses to answer questions which are judged likely to cause confabulations. Results are an average over five datasets, with individual metrics provided in the Supplementary Information.

Thanks
