Table 1: Answers to the question "What is the capital of France?" (a) When all generations from the model mean different things, semantic clustering has no effect—the entropy and semantic entropy are identical. (b) When some of the answers are semantically equivalent ("Paris" and "It's Paris") the semantic entropy does a better job of capturing the actually low uncertainty.

(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	0.94	0.94

(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	0.94	0.33

relation is any reflexive, symmetric, and transitive relation, and that any equivalence relation on a set corresponds to a set of equivalence classes. Each semantic equivalence class corresponds to one possible meaning that our text can have. That is, for the space of semantic equivalence classes $\mathcal{C}$ the sentences in the set $c \in \mathcal{C}$ all share a meaning such that $\forall s, s' \in c : E(s, s')$.

Ordinarily, large language models produce conditional distributions over tokens and their resulting sequences. That is, the probability of the sequence conditioned on the context comes from conditional token probabilities $p(\mathbf{s} \mid x) = \prod_i p(s_i \mid s_{<i}, x)$. Instead, we focus on the probability of the model generating any sequence that shares some meaning. This can be written as

$$p(c \mid x) = \sum_{\mathbf{s}\in c} p(\mathbf{s} \mid x) = \sum_{\mathbf{s}\in c} \prod_i p(s_i \mid s_{<i}, x). \tag{2}$$

Formally, this treats the output random variable whose event-space is $\mathcal{C}$, a sub-σ-algebra of the standard event-space $\mathcal{S}$.

3.2 Sampling the extremely high-dimensional language-space

Recall from Eq. (1) that estimating predictive entropy requires taking an expectation in output-space. However, the output-space of natural language has $\mathcal{O}(|\mathcal{T}|^N)$ dimensions. Moreover, while we can sample from our autoregressive token-model, we lack a normalized probability density function over sentences. The expectation must be approximated by Monte Carlo integration—sampling a finite set of sentences from the output distribution and averaging their likelihoods to compute the entropy. For entropies the average is dominated by low-probability sentences (whose logs are large and negative) making Monte Carlo integration difficult (Mackay, 2003).

3.3 Variable length generations

Sentences of natural language have different lengths. As is widely noted (Murray & Chiang, 2018) and especially in the context of NLG uncertainty by Malinin & Gales (2020), in expectation longer sequences have lower joint likelihoods because of the conditional independence of the token probabilities. The joint likelihood of a sequence of length N shrinks exponentially in N. Its negative log-probability therefore grows linearly in N, so longer sentences tend to contribute more to entropy.

We therefore interpret length-normalising the log-probabilities when estimating the entropy as asserting that the expected uncertainty of generations is independent of sentence length. Sometimes, this is approximately valid. Other times, longer sentences may well be usually more uncertain (e.g., when the goal is to exactly match a typically short reference answer, such as for TriviaQA). In these cases, the advantages of length-normalisation become less clear-cut, as we show empirically in Section 6.1. This offers some guidance *a priori* on cases when length-normalisation is appropriate.

4 Semantic Uncertainty

We have introduced the idea that uncertainty over *meanings* is more important for most situations than uncertainty over the exact tokens used to express those meanings. Our method examines uncertainty in meaning-space—the entropy of the random variable representing the output distribution in the semantic event-space. This is in contrast to entropy in the usual token event-space. To do this we introduce a novel algorithm for estimating the semantic equivalence relation as well as a novel uncertainty estimation algorithm for semantic entropy. At a high level this involves three steps:

1. **Generation:** Sample M sequences $\{s^{(1)}, \ldots, s^{(M)}\}$ 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.

Step 1: Generating a set of answers from the model

First we sample M sequences $\{s^{(1)}, \ldots, s^{(M)}\}$ which we will use later to estimate the uncertainty. These sequences must be sampled according to the distribution $p(\mathbf{s} \mid x)$. In this paper, we sample these sequences only from a *single* model using either multinomial sampling or multinomial beam sampling. We show in Section 6.2, that the choice of sampling temperature and sampling method can have a significant impact on the performance of both our method and the baselines. Unlike Malinin & Gales (2020), we do not use an ensemble of models. Ensembling would probably improve performance, but the cost of training multiple independent foundation models is often prohibitive.

Step 2: Clustering by semantic equivalence

In Section 3.1, we formalised semantic equivalence by introducing the semantic equivalence relation, $E(\cdot, \cdot)$, which induces semantic equivalence classes which are sets of sequences that share a meaning. Recall that the equivalence class c is a set of sequences $\mathbf{s}$ such that $\forall s, s' \in c : E(s, s')$. We operationalise $E(\cdot, \cdot)$ using the idea of bi-directional entailment. A sequence, $\mathbf{s}$, means the same thing as a second sequence, $\mathbf{s}'$, if and only if they entail (i.e. logically imply) each other. E.g., "The capital of France is Paris." entails "Paris is the capital of France." because they mean the same thing.

Importantly, we require that the sequences mean the same thing with respect to the context—key meaning is sometimes contained within the context. For example, "Paris." does not entail "The capital of France is Paris." because "Paris." is not a declarative sentence without context. But within the context of the question, the one-word answer does entail the fuller answer.

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, $\mathbf{s}$ and $\mathbf{s}'$, we detect whether it is possible to infer the concatenation of the context and $\mathbf{s}$ from the concatenation of the context and $\mathbf{s}'$ 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.

Computational cost. The bidirectional equivalence algorithm is combinatorially complex in M, it requires $\binom{M}{2}$-many comparisons in the worst-case. In practice, however, the computational cost is small compared to the cost of generating sequences. First, as we show in Section 6.2, $M < 20$ is often sufficient for good uncertainty. Second, because the Deberta-large model is so much smaller than the main language model (1.5B parameters), each pair comparison is much faster than generating even one token from the main model. Third, because semantic equivalence is transitive we only need to compare one member of each equivalence class to remaining sequences (see Algorithm 1). Additionally, the number of semantic clusters in our tasks is empirically quite low, see Table 2.

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

$$SE(x) = -\sum_{c} p(c \mid x) \log p(c \mid x) = -\sum_{c} \left(\Big(\sum_{\mathbf{s} \in c} p(\mathbf{s} \mid x) \Big) \log \Big[\sum_{\mathbf{s} \in c} p(\mathbf{s} \mid x) \Big] \right). \tag{3}$$

We do not have access to every possible meaning-class c. Instead, we can only sample c from the sequence-generating distribution induced by the model. To handle this, we estimate the expectation in Eq. (3) using Monte Carlo integration over the semantic equivalence classes C from Algorithm 1

$$SE(x) \approx -|C|^{-1} \sum_{i=1}^{|C|} \log p(C_i \mid x). \tag{4}$$

This is an unbiased estimator of the entropy in Eq. (3). In addition, in some cases we use length-normalisation as described in Section 3.3.

4.1 How the semantic entropy addresses the challenges of NLG

The main inspiration of semantic entropy is to address the semantic invariance of natural language head-on by converting the problem of uncertainty estimation into meaning-space. In addition, semantic entropy goes some way towards addressing unequal token importance. Generations whose meanings are the same but differ on unimportant tokens will be added together, which we expect will reduce the effect of the likelihoods of unimportant tokens although we do not demonstrate this empirically. However, this challenge is only partially addressed: semantic entropy will still pay too much attention to non-keyword likelihoods. This is one area where supervised language-model-based uncertainty tools (Lin et al., 2022a; Kadavath et al., 2022) might enable future improvements that handle this challenge better. We address the challenges of sampling and variable-length generation using specific details of our sampling procedure in Section 4.

5 Related Work

Prior work on uncertainty in foundation models for NLP has largely focused on the calibration of *classifiers* (Jiang et al., 2021; Desai & Durrett, 2020) and text regressors (Glushkova et al., 2021; Wang et al., 2022). These settings, are analogous to classification or regression settings in other modalities like vision, and conventional uncertainty measures like MC dropout or Deep Ensembles can be applied without modification (see Section 2 for a discussion of uncertainty in deep learning in general). As we argue in Section 3, generative natural language poses important further challenges. Jiang et al. (2021) do examine calibration in generative question answering and find only a weak correlation between the log-likelihood models assign to their answer and the answer's correctness. In Section 6 we explain however why semantic equivalence in natural language makes calibration a problematic evaluation for generative language models. Reliable uncertainty can be useful on downstream tasks such as graph semantic parsing (Lin et al., 2022b).

Some research has addressed uncertainty or calibration in NLG either by prompting the models to evaluate their own generations or by fine-tuning the generating model to predict its uncertainty (Mielke et al., 2020; Lin et al., 2022a; Kadavath et al., 2022). These methods need further training and supervision. Because they need additional training and supervision, they are hard to reproduce, expensive to create, and have been shown to be sensitive to distribution shift. For example, we were unable to implement one proposal by Kadavath et al. (2022) to train a language model to directly predict confidence due to hardware limitations. Our unsupervised method which uses models 'off-the-shelf' avoids these limitations.

Many of the issues that make probabilistic uncertainty estimation in NLG difficult also make automatic evaluation of NLG difficult. Ott et al. (2018), for instance, study how the performance of machine translation models suffers because one sentence can be translated in multiple ways. Similarly, Sai et al. (2022) discuss how paraphrase detection can be used to evaluate NLG and other related methods might transfer to uncertainty estimation.

Automatic paraphrase identification can be based on comparing lexical features of two given sequences (Fernando & Stevenson, 2008; Issa et al., 2018) or on measuring the similarity between the embeddings of the two sequences (Yu et al., 2014; Socher et al., 2011). Recently, however, SotA paraphrase identification approaches have primarily used BERT-based models to classify pairs of sequences into the classes `paraphrases` and `not paraphrases` (He et al., 2020b; Tay et al., 2021). The idea of formalising semantic equivalence via textual entailment has a long history in linguistics (Culicover, 1968) and NLP (Padó et al., 2009; Androutsopoulos & Malakasiotis, 2010). Transformer-based paraphrase detection models such as EFL (Wang et al., 2021) achieve SotA performance on paraphrase detection benchmarks such as Quora Question Pairs Wang et al. (2017).