LM-Polygraph: Uncertainty Estimation for Language Models

Ekaterina Fadeeva3,5 ♢ Roman Vashurin2 ♢ Akim Tsvigun5,6,7 ♢ Artem Vazhentsev3,4 ♢ Sergey Petrakov3 ♢ Kirill Fedyanin2 ♢ Daniil Vasilyev5 ♢ Elizaveta Goncharova4,5,6
Alexander Panchenko3,4 ♢ Maxim Panov1 ♢ Timothy Baldwin1,8 ♢ Artem Shelmanov1

1 MBZUAI 2 TII 3 Center for Artificial Intelligence Technology 4 AIRI 5 HSE University 6 AI Center NUST MISIS 7 Semrush 8 The University of Melbourne
{ekaterina.fadeeva, sergey.petrakov}@skol.tech {roman.vashurin, kirill.fedyanin}@tii.ae
{akim.tsvigun@semrush.com, vazhentsev, panchenko, goncharova}@airi.net
{maxim.panov, timothy.baldwin, artem.shelmanov}@mbzuai.ac.ae

Abstract

Recent advancements in the capabilities of large language models (LLMs) have paved the way for a myriad of groundbreaking applications in various fields. However, a significant challenge arises as these models often “hallucinate”, i.e., fabricate facts without providing users an apparent means to discern the veracity of their statements. Uncertainty estimation (UE) methods are one path to safer, more responsible, and more effective use of LLMs. However, to date, research on UE methods for LLMs has been focused primarily on theoretical rather than engineering contributions. In this work, we tackle this issue by introducing LM-Polygraph, a framework with implementations of a battery of state-of-the-art UE methods for LLMs in text generation tasks, with unified program interfaces in Python. Additionally, it introduces an extendable benchmark for consistent evaluation of UE techniques by researchers, and a demo web application that enriches the standard chat dialog with confidence scores, empowering end-users to discern unreliable responses. LM-Polygraph is compatible with the most recent LLMs, including BLOOMz, LLaMA-2, ChatGPT, and GPT-4, and is designed to support future releases of similarly-styled LMs.

1 Introduction

Large language models (LLMs) have demonstrated remarkable performance across a variety of text generation tasks. Instruction fine-tuning and reinforcement learning from human feedback (RLHF) have brought the zero-shot performance of these models to a new level (Ouyang et al., 2022). However, the capabilities of LLMs, despite their profound power and complexity, are inherently constrained. Limitations arise from the finite nature of the training data and the model’s intrinsic memorization and reasoning capacities. Hence, their utility is bounded by the depth and breadth of the knowledge they embed.

Due to their training objectives, even when the embedded knowledge of an LLM on a given topic is limited, it tends to be over-eager to respond to a prompt, sometimes generating misleading or entirely erroneous output. This dangerous behavior of attempting to appease the user with plausible-sounding but potentially false information is known as “hallucination” (Xiao and Wang, 2021; Dziri et al., 2022). It poses a significant challenge when deploying LLMs in practical applications.

There are several well-known approaches to censoring LLM outputs, including: filtering with stop-word lists, post-processing with classifiers (Xu et al., 2023), rewriting of toxic outputs (Logacheva et al., 2022), and longer fine-tuning with RLHF. However, these approaches cannot be relied on to completely resolve hallucinations. Since LLMs are natural (if “unintentional”) liars, we propose LM-Polygraph — a program framework that, similar to a human polygraph, leverages various hidden signals to reveal when one should not trust the subject. In particular, LM-Polygraph provides a comprehensive collection of uncertainty estimation (UE) techniques for LLMs in text generation tasks.

Uncertainty estimation refers to the process of quantifying the degree of confidence in the predictions made by a machine learning model. For classification and regression tasks, there is a well-developed battery of methods (Gal, 2016). There has also been a surge of work investigating UE, particularly in text classification and regression in conjunction with encoder-only LMs such as BERT (Zhang et al., 2019; He et al., 2020; Shelmanov et al., 2021; Xin et al., 2021; Vazhentsev et al., 2022; Kotelevskii et al., 2022; Wang et al., 2022; Kuzmin et al., 2023). However, UE for sequence generation tasks, including text generation,
is a much more complex problem. To quantify the uncertainty of the whole sequence, we have to aggregate uncertainties of many individual token predictions and deal with non-trivial sampling and pruning techniques like beam search. Contrary to classification tasks where the number of possible prediction options is finite, in text generation, the number of possible predictions is infinite or exponential in vocabulary size, complicating the estimation of probabilities and information-based scores. Finally, a natural language text is not a simple sum of its tokens; it is a nuanced interpolation of context, semantics, and grammar, so two texts can have very diverse surface forms but similar meanings, which should be taken into account during the UE process.

Several recent studies have delved into developing UE methods for LMs in text generation tasks (Malinin and Gales, 2021; van der Poel et al., 2022; Kuhn et al., 2023; Ren et al., 2023; Vazhentsev et al., 2023b; Lin et al., 2023). However, the current landscape of this research is quite fragmented with many non-comparable or even concurrent studies, which makes it challenging to consolidate the findings and draw holistic conclusions.

In this work, with the development of LM-Polygraph, we strive to bridge these disparate research efforts, fostering more cohesion and synergy in the field. We envision a framework that consolidates the scattered UE techniques within unified frameworks in Python, provides an extendable evaluation benchmark, and offers tools to integrate uncertainty quantification in standard LLM pipelines seamlessly. This endeavor will not only make the journey less challenging for individual researchers and developers but also set the stage for more robust, reliable, and trustworthy LLM deployments for end-users.

Our contributions are as follows:

- We provide a comprehensive framework that implement state-of-the-art methods for UE of LM predictions. We also provide the ability to combine multiple uncertainty scores together as suggested by Ren et al. (2023); Vazhentsev et al. (2023a).
- We create a tool that enriches standard LLM chat capabilities with uncertainty scores for model outputs. The tool can potentially be used by end-users to determine whether the answers of language models are reliable or not, and by researchers to develop novel UE techniques for LMs in text generation tasks.
- We construct an easy-to-extend benchmark for UE methods in text generation tasks and provide reference experimental results for implemented UE techniques.

2 Python Library

LM-Polygraph implements a set of state-of-the-art UE techniques for LLMs with unified program interfaces in Python. It is compatible with models from the Huggingface library and is tested with recent public-domain LLMs such as BLOOMz (Scao et al., 2022; Yong et al., 2023), Dolly v2 (Conover et al., 2023), Alpaca (Taori et al., 2023), LLaMA-2 (Touvron et al., 2023), and Flan-T5 (Chung et al., 2022). The framework supports both conditional models with a seq2seq architecture and unconditional decoder-only LMs. Figure 1 contains a code example of LM-Polygraph with BLOOMz-3B for UE in open-domain question answering. Some methods that do not require access to the model itself or its logits could be used in conjunction with web-hosted LLMs like ChatGPT or GPT-4 through APIs. We provide a program wrapper for integration with popular online services.

3 Uncertainty Estimation Methods

Here, we summarize UE methods implemented in LM-Polygraph, as listed in Table 1.

There are two major technique types: white-box and black-box. The white-box methods require access to logits, internal layer outputs, or the LM itself. The black-box methods require access only to the generated texts, and can easily be integrated with third-party online services like OpenAI LM API. We note that the methods differ by computational requirements: some techniques pose high computational requirements: some techniques pose high

```python
from lm_polygraph import estimate_uncertainty
from lm_polygraph import WhiteboxModel
from lm_polygraph.estimators import *
model = WhiteboxModel.from_pretrained(
    "bigscience/bloomz-3b",
    device="cuda:0",
) 
ue_method = MeanPointwiseMutualInformation()
input_text = "Who is George Bush?"
estimate_uncertainty(model, ue_method, input_text)
```

Figure 1: Code example of how LM predictions could be enriched with uncertainty scores using LM-Polygraph.
We start the discussion of white-box techniques with a focus on white-box methods, which leverage the diversity of output predictions made by multiple slightly different versions of models under slightly different conditions. Let us assume that $M$ models are available with parameters $\theta_i$, $i = 1, \ldots, M$. These parameters can be obtained via independent training of models. Then one can use token $P(y_l | y_{<l}, x, \theta_i)$ and sequence $P(y | x, \theta_i)$ probabilities to compute various metrics such as mutual information that measures the discrepancy between model predictions.

**Density-based methods** leverage latent representations of instances and construct a probability density on top of them. Usually, these methods approximate training data distribution with the help of one or multiple Gaussian distributions. They can provide a probability or an unnormalized score that determines how likely instances belong to the training data distribution. Therefore, they are good at spotting out-of-distribution (OOD) instances (Vazhentsev et al., 2023b). Several variations of these methods have been proposed in the literature (Lee et al., 2018; Yoo et al., 2022; Ren et al., 2023; Kotelevskii et al., 2022).

The primary advantage of these methods is that they are computationally efficient: they do not need much time for additional model inference, and memory overhead for storing additional parameters is minimal. The drawback is that these methods require access to the model’s training data to fit auxiliary models like Gaussians (e.g., the Mahalanobis Distance method requires constructing computational or memory overheads, e.g., due to repeated inference, making them less suitable for practical usage. The application of some methods also can be hindered by the need for access to the model training data.

Let us consider the input sequence $x$ and the output sequence $y \in \mathcal{Y}$ of length $L$, where $\mathcal{Y}$ is a set of all possible output sequences. Then the probability of an output sequence given an input sequence for probabilistic autoregressive language models is given by:

$$P(y | x, \theta) = \prod_{l=1}^{L} P(y_l | y_{<l}, x, \theta),$$

where the distribution of each $y_l$ is conditioned on all the previous tokens in a sequence $y_{<l} = \{y_1, \ldots, y_{l-1}\}$, and $\theta$ denotes the parameters of the model.

### 3.1 White-box Methods

We start the discussion of white-box techniques from information-based methods. These techniques are based on token $P(y_l | y_{<l}, x, \theta)$ and sequence $P(y | x, \theta)$ probabilities obtained from a single model prediction. The notable example is entropy, which can be calculated on the token or sequence level. The benefits of information-based methods are that they are cheap to compute and simple to implement. However, the quality of these methods is usually relatively low, so they are considered as baselines. Some domain-specific methods were recently proposed in an attempt to improve over standard information-based approaches, such as semantic entropy (Kuhn et al., 2023).
Figure 2: User interface of the demo. A user can interact with an LLM as with any other chat service, but in LM-Polygraph the user also sees the confidence of the model answers. It is possible to specify various UE techniques and various models, including ChatGPT.

data centroids and covariance matrices). These methods are also known to capture only epistemic uncertainty. Therefore, they might not be perfect for selective generation as they cannot be used to spot ambiguous in-domain instances.

Finally, we also combine information-based and density-based methods as suggested by Vazhentsev et al. (2023a) and Ren et al. (2023). More specifically, we implement the hybrid uncertainty quantification (HUQ) method (Vazhentsev et al., 2023a) that performs a ranking-based aggregation and leverages strengths of both information-based methods that detect ambiguous instances and density-based methods that detect OOD instances.

Directly asking the model to validate its answer is another option for UE (Kadavath et al., 2022). In this method, one asks models first to propose answers and then to evaluate the probability $P(\text{True})$ that their answers are correct. Kadavath et al. (2022) show that it achieves reasonable performance on a variety of tasks, including question-answering. We note that this method requires inference of a model twice: the first to generate an answer, and the second for processing its own output. Even though the second inference is usually faster than the first one, it still takes considerable time for computation.

3.2 Black-box Methods

In contemporary models, there are instances where the model’s architecture and hidden states are unavailable or there is no access to logits during response generation. Nevertheless, a whole class of black box methods only needs to access the model’s response. Within the scope of this paper, we consider several approaches of this type that have performed well in other studies (Fomicheva et al., 2020; Kuhn et al., 2023; Lin et al., 2023). We focus on Lexical Similarity, Number of Semantic Sets, Sum of Eigenvalues of the Graph Laplacian, Degree Matrix, and Eccentricity. We use the same methodological approach as the authors of
the work (Lin et al., 2023):
• Obtain $K$ responses $y_1, \ldots, y_K$ for a particular input $x$.
• Compute $K \times K$ similarity matrix $S$ between responses, where $S_{ij} = s(y_i, y_j)$ for some similarity score $s$ (Natural Language Inference score or Jaccard score).
• Based on the similarity matrix $S$, we compute the final uncertainty score.

Thus, the idea of the methods is to analyze the similarity matrix and aggregate the information to compute the uncertainty score.

4 Demo

We constructed a demo application that can be used to interact with LLMs and also see confidence scores of model answers (see Figure 2). A user specifies a UE method and a language model from a number of publicly-available LLMs with up to 13B parameters, e.g., BLOOMz, Vicuna, and LLaMA-2. There is also the ability to communicate with LLMs deployed as web services such as ChatGPT or GPT-4 and obtain their uncertainty scores based on black-box techniques. For these models, a user should provide an API key.

This demo application is potentially helpful for both end-users and researchers. For end-users, it extends the standard AI assistant interface with information about whether it is reasonable to trust a model answer. Researchers could use this tool for qualitative analysis of various UE methods and LLM responses.

5 Evaluation Benchmark

LM-Polygraph provides a vast evaluation benchmark. It contains a script for running one or multiple experiments with UE techniques, implemented as Python modules. This feature allows the user to easily extend the set of available methods and evaluate novel UE techniques in a unified manner. Using this benchmark, we have conducted experiments with most methods implemented in LM-Polygraph. Below, we provide experimental details.

Datasets. We experiment with three text generation tasks: machine translation (MT), text summarization (TS), and question answering (QA). For each task, we use two widely-used datasets: WMT-14 German to English and WMT-14 French to English (Bojar et al., 2014) for MT, XSum (Narayan et al., 2018) and AESLC (Zhang and Tetreault, 2019) for TS, and CoQA (Reddy et al., 2019) and bAbI QA (Dodge et al., 2016) for QA. Dataset statistics are presented in Appendix D.

Models. We conducted experiments with the Vicuna-v1.5-7B (Zheng et al., 2023) and Llama-v2-7B (Touvron et al., 2023) models. The generation hyperparameters are provided in Appendix B.

Metrics. We focus on the task of selective generation (Ren et al., 2023) where we “rejecting” generated sequences due to low quality based on uncertainty scores. Rejecting means that we do not use the model output, and the corresponding queries are processed differently: they could be further reprocessed manually or sent to a more advanced LLM. Following previous work on UE in text generation (Malinin and Gales, 2021; Vazhentsev et al., 2022), we compare the methods using the Prediction Rejection Ratio (PRR) metric (Malinin et al., 2017).

Consider a test dataset $D = \{(x_i, y_i)\}$. Let $f(x_i)$ be the output generated by an LLM and $U(x_i)$ be the uncertainty score of a prediction. The prediction rejection (PR) curve indicates the dependence of the average quality $Q(f(x_i), y_i)$ of the covered instances from the uncertainty rate $a$ used for rejection, in ascending order. We use ROUGE-L and BERTScore (Zhang et al., 2020) as text quality metrics $Q(f(x_i), y_i)$. Finally, PRR computes the ratio of the area $AUCPR_{unc}$ between the PR curve for the uncertainty estimates and random estimates and the area $AUCPR_{oracle}$ between the oracle and random estimates:

$$ PRR = \frac{AUCPR_{unc}}{AUCPR_{oracle}} $$

Higher PRR values indicate better quality of selective generation.

6 Experimental Results

Tables 2 and 3 present the results for Vicuna-v1.5-7b and Llama-v2-7b correspondingly.

For both models, the better performance is usually demonstrated by the white-box methods based on information-theoretic concepts (first 8 rows of the table). These methods are in general also easy to implement and computationally lightweight, with the notable exceptions of Semantic Entropy, Monte Carlo Sequence Entropy, and Monte Carlo Normalized Sequence Entropy, which require sampling from the model several times to obtain uncertainty scores.
When working with LLMs as web services, usually there is no access to full posterior distributions over tokens, therefore, only black-box methods could be used. Among this group of approaches, the best average performance is achieved by Centrality for Vicuna. For LLaMA, there is no clear advantage for any of the methods considered.

Overall, we see that absolute values for all evaluated methods, models, and datasets are far away from perfect. Low performance of current methods is especially evident on more complicated tasks such as XSum and WMT14. Our experimental results demonstrate that the task of selective generation is not close to be solved. This once again underlines the importance of further research and development of efficient uncertainty estimation techniques for generative language models.

## 7 Conclusion

As the community strives to advance the potential of LLMs, it is critical to be mindful about dangers of their uncontrolled usage. In this work, we propose a tool for making the application of LLMs safer. Enriching model predictions with uncertainty scores helps users and developers to be informed about these risks, encouraging healthy skepticism towards certain outputs generated by these models.

We plan to further extend our framework with implementations of new UE methods that emerge in the future. We hope that our work will foster the development of techniques to detect and mitigate LLM hallucinations, which we believe is a key to unlocking the safe, responsible, and effective use of LLMs in real-world applications.
Limitations

We have tried to be as comprehensive as possible with our collection of UE methods. However, we omit several techniques that have not demonstrated strong performance in previous work, do not have a strong theoretical motivation, or are similar to other implemented techniques.

We note that comprehensive evaluation of UE methods is an open research question. LM-Polygraph makes the first steps to systematize, and provide interfaces and tools for testing UE techniques in a unified manner. However, we believe that the number of tasks and datasets should be extended in the future.

When running the demo, we cannot provide an access to the biggest and the most powerful public LLMs, because running them is prohibitively expensive. Nevertheless, a user can access models such as ChatGPT by providing an API access key.

LM-Polygraph supports common application program interfaces used by modern LLMs. However, it is possible that certain modifications will be required to support future releases of LLMs.

At the moment of writing, LM-polygraph provides valid uncertainty estimates only for model outputs in English language. This is due to the fact that most generation quality metrics implemented are based off English-specific implementations and non-multilingual models. We plan to alleviate this limitation by allowing the user to easily employ custom quality metrics and scoring models.

Ethics Statement

We conducted all experiments on publicly-available datasets that have been leveraged in various previous work on uncertainty estimation of LLMs.

While training data for most LLMs, such as BLOOMz, was selected to contain little or no abusive text content, such models can still potentially output harmful textual content. Techniques investigated in our work estimate certainty of an LM output to “censor” its output, and model debiasing is an orthogonal direction to our line of work. These additional methods can and perhaps should be combined in real production LLM deployments. We hope that our framework contributes to safer and more reliable usage of language models.

Acknowledgements

We thank the anonymous reviewers for their insightful feedback towards improving this paper.


Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilic, Daniel Hesselow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon,


A Methods Description

Here, we summarize UE methods implemented in LM-Polygraph; see also Table 1.

A.1 White-box Methods

A.1.1 Information-based methods

Maximum sequence probability score simply leverages the probability of the most likely sequence generation: \( \text{MSP}(y \mid x, \theta) = 1 - P(y \mid x, \theta) \).

Length-normalized log probability computes the average negative log probability of generated tokens. If the score is exponentiated it corresponds to \textit{perplexity}. The resulting quantity is computed as

\[
P(y, x; \theta) = \exp\left\{ -\frac{1}{L} \log P(y \mid x, \theta) \right\},
\]

while it is convenient also to denote length-normalized sequence probability by \( \hat{P}(y \mid x, \theta) = \exp\left\{ \frac{1}{L} \log P(y \mid x, \theta) \right\} \).

We also provide the \textit{mean token entropy}, where we simply average entropy of each individual token in the generated sequence:

\[
\mathcal{H}_{T}(y; x; \theta) = \frac{1}{L} \sum_{l=1}^{L} \mathcal{H}(y_{l} \mid y_{<l}, x, \theta),
\]

where \( \mathcal{H}(y_{l} \mid y_{<l}, x, \theta) \) is an entropy of the token distribution \( P(y_{l} \mid y_{<l}, x, \theta) \).

The other possibility to compute entropy-based uncertainty measure is to compute it on the level of whole sequences via \( E \left[- \log P(y \mid x, \theta)\right] \), where expectation is taken over the sequences \( y \) randomly generated from the distribution \( P(y \mid x, \theta) \). In practice, one needs to use Monte-Carlo integration, i.e. generate several sequences \( y^{(k)}, k = 1, \ldots, K \) via random sampling and compute the resulting Monte Carlo Sequence Entropy:

\[
\mathcal{H}_{S}(x; \theta) = -\frac{1}{K} \sum_{k=1}^{K} \log P(y^{(k)} \mid x, \theta). \tag{3}
\]

The same procedure can be done by substituting \( P(y^{(k)} \mid x, \theta) \) with its length-normalized version \( \hat{P}(y^{(k)} \mid x, \theta) \) leading to a more reliable uncertainty measure in some applications.

Another entropy-based uncertainty measure is \textit{Semantic Entropy} proposed by Kuhn et al. (2023). The method aims to deal with the generated sequences that have similar meaning while having different probabilities according to the model, which can significantly affect the resulting entropy value (3). The idea is to cluster generated sequences \( y^{(k)}, k = 1, \ldots, K \) into several semantically homogeneous clusters \( c_{m}, m = 1, \ldots, M \) with \( M \leq K \) with bi-directional entailment algorithm and average the sequence probabilities within the clusters. The resulting estimate of entropy is given by the following formula:

\[
\text{SE}(x; \theta) = -\sum_{m=1}^{M} \hat{P}_{m}(x; \theta) \log \hat{P}_{m}(x; \theta),
\]

where \( \hat{P}_{m}(x; \theta) = \frac{1}{|c_{m}|} \sum_{y \in c_{m}} P(y \mid x, \theta) \).

Finally, one can consider negative mean \textit{Pointwise Mutual Information} (PMI; Takayama and Arase (2019)) which is given by

\[
\text{PMI}(y, x; \theta) = \frac{1}{L} \sum_{l=1}^{L} \log \frac{P(y_{l} \mid y_{<l}, x, \theta)}{P(y_{l} \mid y_{<l})}.
\]

This method was extended in (van der Poel et al., 2022) by considering only those marginal probabilities for which the entropy of the conditional distribution is above certain threshold: \( \mathcal{H}(y_{l} \mid y_{<l}, x, \theta) \geq \tau \). It leads to the negative mean \textit{Conditional Pointwise Mutual Information} (CPMI) measure that is given by:

\[
\text{CPMI}(y, x; \theta) = -\frac{1}{L} \sum_{l=1}^{L} \log P(y_{l} \mid y_{<l}, x, \theta) + \frac{\lambda}{L} \sum_{l: \mathcal{H}(y_{l} \mid y_{<l}, x, \theta) \geq \tau} \log P(y_{l} \mid y_{<l}, \theta),
\]

where \( \lambda > 0 \) is another tunable parameter.

A.1.2 Ensemble-based methods

For the ensembling on a sequence level, we consider two uncertainty measures: total uncertainty measured via average sequence probability \( \frac{1}{M} \sum_{i=1}^{M} \hat{P}(y \mid x, \theta_{i}) \):

\[
\text{MSP}_{S}(y, x) = 1 - \hat{P}(y \mid x) \tag{4}
\]

and

\[
\text{M}_{S}(y, x) = \frac{1}{M} \sum_{i=1}^{M} \log \frac{P(y \mid x)}{\hat{P}(y \mid x, \theta_{i})}, \tag{5}
\]

which is known as \textit{reverse mutual information} (RMI).

Next we discuss token level uncertainty measures and start with a total uncertainty estimate via entropy:

\[
\mathcal{H}_{T}(y, x) = \sum_{l=1}^{L} \mathcal{H}(y_{l} \mid y_{<l}, x), \tag{6}
\]
where $H(y_l | y_{<l}, x)$ is an entropy of the token distribution $P(y_l | y_{<l}, x) = \frac{1}{M} \sum_{i=1}^{M} P(y_l | y_{<l}, x, \theta_i)$.

Additionally, for the ensemble one can compute the variety of other token level uncertainty measures including average entropy of ensemble members (also known as Data Uncertainty):

$$D(y_l | y_{<l}, x) = \frac{1}{M} \sum_{i=1}^{M} H(y_l | y_{<l}, x, \theta_i),$$

Mutual Information (MI):

$$I(y_l | y_{<l}, x) = H(y_l | y_{<l}, x) - D(y_l | y_{<l}, x)$$

and Expected Pairwise KL Divergence (EPKL):

$$K(y_l | y_{<l}, x) = \left( \frac{M}{2} \right)^{-1} \cdot \sum_{i \neq j} KL(P(y_l | y_{<l}, x, \theta_i) || P(y_l | y_{<l}, x, \theta_j)),$$

where $KL(P || Q)$ refers to a KL-divergence between distributions $P$ and $Q$.

Finally, Reverse Mutual Information (RMI) also can be computed on the token level via a simple equation

$$M(y_l | y_{<l}, x) = K(y_l | y_{<l}, x) - I(y_l | y_{<l}, x).$$

The resulting token-level uncertainties computed via Data Uncertainty, MI, EPKL and RMI can be plugged-in in equation (6) on the place of entropy leading to corresponding sequence level uncertainty estimates.

### A.1.3 Density-based Methods

Let $h(x)$ be a hidden representation of an instance $x$. The Mahalanobis Distance (MD; Lee et al. (2018)) method fits a Gaussian centered at the training data centroid $\mu$ with an empirical covariance matrix $\Sigma$. The uncertainty score is the Mahalanobis distance between $h(x)$ and $\mu$:

$$MD(x) = (h(x) - \mu)^T \Sigma^{-1} (h(x) - \mu).$$

We suggest using the last hidden state of the encoder averaged over non-padding tokens or the last hidden state of the decoder averaged over all generated tokens as $h(x)$.

The Robust Density Estimation (RDE; Yoo et al. (2022)) method improves over MD by reducing the dimensionality of $h(x)$ via PCA decomposition. Additionally, computing of the covariance matrix $\Sigma$ for each individual class is done by using the Minimum Covariance Determinant estimation (Rousseeuw, 1984). The uncertainty score is computed as the Mahalanobis distance between but in the space of reduced dimensionality.

Ren et al. (2023) showed that it might be useful to adjust the Mahalanobis distance score by subtracting from it the other Mahalanobis distance $MD_0(x)$ computed for some large general purpose dataset covering many domains like C4 (Raffel et al., 2020). The resulting resulting Relative Mahalanobis Distance score is

$$RMD(x) = MD(x) - MD_0(x).$$

### A.2 Black-box Methods

In this work, we follow Lin et al. (2023) and consider two approaches to compute the similarity for the generated responses. The first one is Jaccard similarity:

$$s(y, y') = \frac{|y \cap y'|}{|y \cup y'|},$$

where the sequences $y$ and $y'$ are considered just as sets of words.

The other similarity measure considered is Natural Language Index (NLI) which employs a classification model to identify whether two responses are similar. We follow Kuhn et al. (2023) and use the DeBERTa-large model (He et al., 2021) that, for each pair of input sequences, provides two probabilities: $\hat{p}^\text{entail}(y, y')$ that measures the degree of entailment between the sequences and $\hat{p}^\text{contra}(y, y')$ that measures the contradiction between them. Then one can use $s^\text{entail}(y, y') = \hat{p}^\text{entail}(y, y')$ or $s^\text{contra}(y, y') = 1 - \hat{p}^\text{contra}(y, y')$ as a measure of similarity between sequences $y$ and $y'$.

**Number of Semantic Sets** illustrates whether answers are semantically equivalent. We adopt an iterative approach by sequentially examining responses from the first to the last while making pairwise comparisons between them (each pair has indexes $j_1$ and $j_2$, $j_2 > j_1$). The number of semantic sets initially equals the total number of generated answers $K$. If the condition $\hat{p}^\text{entail}(y_{j_1}, y_{j_2}) > \hat{p}^\text{contra}(y_{j_1}, y_{j_2})$ and $\hat{p}^\text{entail}(y_{j_2}, y_{j_1}) > \hat{p}^\text{contra}(y_{j_2}, y_{j_1})$ is fulfilled we put this two sentences into one cluster. The computation is done for all the pairs of answers, and then the resulting number of distinct sets $U_{\text{NumSemSets}}$. 

457
is reported. It is worth noting that a higher number of semantic sets corresponds to an increased level of uncertainty, as it suggests a higher number of diverse semantic interpretations for the answer.

Nonetheless, it is essential to acknowledge a limitation of this measure: it can only take integer values. Additionally, it cannot be assumed that the semantic equivalence derived from the NLI model is always transitive. Consequently, the authors of (Lin et al., 2023) suggest the consideration of a continuous counterpart of this metric. They propose the Sum of Eigenvalues of the Graph Laplacian as a potential alternative approach.

Let’s consider a similarity matrix $S_{j_1,j_2} = (s(y_{j_1}, y_{j_2}) + s(y_{j_2}, y_{j_1}))/2$. Averaging is done to obtain better consistency. Normalized Graph Laplacian of the obtained similarity Matrix $S$ has the following formula $L = I - D^{-1/2} S D^{-1/2}$, where $D$ is a diagonal matrix and $D_{ii} = \sum_{j=1}^{K} S_{ij}$. Consequently, the following formula is derived: $U_{EigV} = \sum_{k=1}^{K} \max(0, 1 - \lambda_k)$. This value is a continuous analogue of $U_{NumSemSets}$. In extreme case if adjacency matrix $S$ is binary these two measures will coincide.

Of course, from a theoretical and practical point of view, $U_{EigV}$ is a much more flexible approach compared to $U_{NumSemSets}$. Still, they have a common disadvantage: they can not provide uncertainty for each answer. However, authors of (Lin et al., 2023) demonstrate that we can take it from Degree Matrix $D$ computed above. The idea is that the total uncertainty of the answers might be measured as a corrected trace of the diagonal matrix $D$ because elements on the diagonal of matrix $D$ are sums of similarities between the given answer and other answers. Thus, it is an average pairwise distance between all answers, and a larger value will indicate larger uncertainty because of the larger distance between answers. The resulting uncertainty measure becomes $U_{Deg} = 1 - \text{trace}(D)/K^2$.

A drawback of previously considered methods is the limited knowledge of the actual embedding space for the different answers since we only have measures of their similarities. Nevertheless, we can overcome this limitation by taking advantage of the inferential capabilities of the graph Laplacian, which makes it easier to obtain the coordinates of the answers. Let us introduce $u_1,\ldots,u_k \in \mathbb{R}^K$ as the eigenvectors of $L$ that correspond to $k$ smallest eigenvalues. We can efficiently construct an informative embedding $v_j = [u_{1,j},\ldots,u_{k,j}]$ for an answer $y_j$. Authors of (Lin et al., 2023) demonstrate that this approach allows the usage of the average distance from the center as an uncertainty metric and to consider the distance of each response from the center as a measure of (negative) confidence. In mathematical terms, the estimates for Eccentricity can be defined as follows: $U_{Ecc} = \|\tilde{v}_1^T,\ldots,\tilde{v}_K^T\|_2$, where $\tilde{v}_j = v_j - \frac{1}{K} \sum_{\ell=1}^{K} v_\ell$.

Last but not least, Lexical Similarity is a measure proposed by (Fomicheva et al., 2020) that computes how similar two words or phrases are in terms of their meaning. Since the original article is dedicated to machine translation, this measure calculates the average similarity score between all pairs of translation hypotheses in a set, using a similarity measure based on the overlap of their lexical items. Different metrics can be used, such as ROUGE-1, ROUGE-2, ROUGE-L, and BLEU. For our task, this measure iterates over all responses and calculates the average score with other answers.
**B Generation Hyperparameters**

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Task</th>
<th>Max Input Length</th>
<th>Generation Length</th>
<th>Temperature</th>
<th>Top-p</th>
<th>Do Sample</th>
<th>Beams</th>
<th>Repetition Penalty</th>
</tr>
</thead>
<tbody>
<tr>
<td>AESLCl</td>
<td>ATS</td>
<td>31</td>
<td>1.0</td>
<td>False</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XSUM</td>
<td>QA</td>
<td>56</td>
<td>1.0</td>
<td>False</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CoQA</td>
<td></td>
<td>20</td>
<td>1.0</td>
<td>False</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bAbiQA</td>
<td></td>
<td>3</td>
<td>1.0</td>
<td>False</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WMT14 De-En</td>
<td></td>
<td>107</td>
<td>1.0</td>
<td>False</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WMT14 Fr-En</td>
<td></td>
<td>107</td>
<td>1.0</td>
<td>False</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4: Text generation hyperparameters for both LLMs Vicuna-v1.5-7b and Llama-2-7b used in the experiments.

Table 4 presents the hyperparameters used for experiments with LLMs Vicuna-v1.5-7b and LLama-2-7b-hf on various datasets and tasks. Maximum length of generated sequence was set for each dataset as the 99th percentile of target sequence length on the respective train set.

**C Text Generation Quality Metrics**

<table>
<thead>
<tr>
<th>AESLCl</th>
<th>XSUM</th>
<th>CoQA</th>
<th>bAbiQA</th>
<th>WMT14 De-En</th>
<th>WMT14 Fr-En</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rouge-L</td>
<td>0.24</td>
<td>0.18</td>
<td>0.29</td>
<td>0.68</td>
<td>0.59</td>
</tr>
<tr>
<td>BERTScore</td>
<td>0.83</td>
<td>0.86</td>
<td>0.85</td>
<td>1.0</td>
<td>0.95</td>
</tr>
</tbody>
</table>

Table 5: Rouge-L↑ and BERTScore↑ for Vicuna v1.5 model for various tasks.

<table>
<thead>
<tr>
<th>AESLCl</th>
<th>XSUM</th>
<th>CoQA</th>
<th>bAbiQA</th>
<th>WMT14 De-En</th>
<th>WMT14 Fr-En</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rouge-L</td>
<td>0.23</td>
<td>0.19</td>
<td>0.51</td>
<td>0.36</td>
<td>0.54</td>
</tr>
<tr>
<td>BERTScore</td>
<td>0.84</td>
<td>0.86</td>
<td>0.91</td>
<td>0.98</td>
<td>0.93</td>
</tr>
</tbody>
</table>

Table 6: Rouge-L↑ and BERTScore↑ for the Llama v2 model for various tasks.

**D Dataset Statistics**

Table 7 illustrates the statistics of the datasets that were used in the experiments. Experiments were conducted using all examples from the test sets of these datasets, while training density-based methods were performed on a random subset of 1000 elements from the train set.

**E Normalization of Uncertainty Estimates in Demo App**

To make uncertainty estimation more intuitive for the end user, directly interacting with the LLM, we perform normalization of various uncertainty estimates. After normalization the output \( UE(x) \) of any uncertainty estimation approach becomes a confidence score \( C(x) \in [0, 1] \subset \mathbb{R} \).

We experimented with several ways of achieving this normalization, including quantile-based approach and simple linear normalization on maximum value obtained from validation dataset. Eventually we
### Table 7: Quantitative Information Regarding the Datasets from Experiments

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Num. instances</th>
<th>Av. document len.</th>
<th>Av. target len.</th>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>NMT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WMT'14</td>
<td>4.51M/3000/3003</td>
<td>19.8/18.3</td>
<td>23.0/21.3</td>
<td>German-to-English</td>
</tr>
<tr>
<td>WMT'14</td>
<td>40.8M/3000/3003</td>
<td>33.5/32.1</td>
<td>29.2/27.0</td>
<td>French-to-English</td>
</tr>
<tr>
<td>ATIS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XSum</td>
<td>2040/11332/11334</td>
<td>454.6</td>
<td>26.1</td>
<td>English</td>
</tr>
<tr>
<td>AESLCC</td>
<td>14436/1960/1906</td>
<td>165.5</td>
<td>6.7</td>
<td>English</td>
</tr>
<tr>
<td>QA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CoQA</td>
<td>7199/500/-</td>
<td>271.4</td>
<td>2.7</td>
<td>English</td>
</tr>
<tr>
<td>bAbiQA</td>
<td>2000/-/200</td>
<td>31.1</td>
<td>1.0</td>
<td>English</td>
</tr>
</tbody>
</table>

Figure 4: Config Example for Question Answering on CoQA Dataset.

performed normalization as a calibration procedure, where normalized confidence score represents expected value of generation quality metric of choice (i.e. RougeL) for a given uncertainty estimate. This expectation is estimated by computing sample averages of quality metric over bins of uncertainty estimates, calculated for some validation dataset. For RougeL metric, the confidence estimate \( C(x_{\text{input}}) \) thus becomes:
\[ C(\text{x}_{\text{input}}) = \frac{1}{|B|} \sum_{x_i,y_i \in B} \text{rougeL}(\hat{y}_i, y_i), \]

where \( \hat{y}_i \) is model output for input \( x_i \), and

\[ B = \{ (x, y) \in D_{\text{calib}} \mid \text{UE}(x) \in [\text{UE}_{\text{min}}, \text{UE}_{\text{max}}) \} \]

is the bin to which uncertainty estimate of the input belongs. The bounds of this bin are selected from the predetermined set of bin boundaries to be the neighboring pair for which condition

\[ \text{UE}_{\text{min}} \leq \text{UE}(\text{x}_{\text{input}}) < \text{UE}_{\text{max}} \]

is satisfied.

This dataset \( D_{\text{calib}} \) is constructed to be representative of different modes of operation of a given model. For this purpose it is constructed as a mixture of several different datasets for different tasks, with different values of relevant statistics, such as input sequence length, typical generated output length etc.

It is obvious that quality of this normalized confidence score depends heavily on the size and diversity of the calibration dataset. In general we consider the problem of translating opaque uncertainty estimates into intuitive absolute confidence scores, that correctly represent likelihood of the generated output being correct and relevant, as an important and complicated task. We leave solving this problem in a more efficient and universal way to the future work.