École normale supérieure Paris-Saclay
Master MVA

ASSESSING UNCERTAINTY IN LLMS: DIFFERENTIATING THEN COMBINING PROMPT-BASED AND PURELY LOGIT-BASED APPROACHES

Mathias Vigouroux

Supervisors:
Yann Chevaleyre, Jamal Atif
Paris-Dauphine University

Referent teacher:
Gabriel Peyré
École Normale Supérieure Paris

15/04/2024 - 15/09/2024

1 Preface

I have always been fascinated by the behaviors of animals and humans. One behavior that I find particularly interesting is anxiety. Indeed, I have encountered people (including myself) who tend to be somewhat anxious. I believe we can agree that the nature of events seems quite random, stochastic even. Thus, trying to figure out how to navigate this variable environment is a challenging task. Taking some distance, the ability to understand that there are more or less risky outcomes and information that you do not have access to is highly linked to metacognition, the ability to reflect on oneself, which itself relies on consciousness [9].

At some point, I tried to understand why people are more or less risk-taking and why they might wrongly estimate the odds of events, either due to nature (genetically) or nurture (learned). This hypothesis could be tested by controlling for genetic factors while predicting behavior using brain imagery data. I then had the opportunity to work on how humans process the variability in the physical measures that might give rise to uncertainty, namely time.

I moved from using naïve statistical tools such as SNP and voxel-based morphometry to slightly more advanced models: Bayesian ones, which learned small Gaussian distributions. As I continued to explore more complex models and simplify the tasks of interest, I decided to dive into models considered to depict the most advanced capacities for reasoning—Large Language Models (LLMs). Faced with these complex models, the behavior of interest had to become even simpler. I thus started to explore how ChatGPT could say, "I do not know."

This question was first discussed with Lorenz Kuhn, now a researcher at OpenAI, who has published two papers specifically on these topics, which will be developed in detail in the rest of this thesis [17], [18].

Some critical and strategic industries (health, defense, law, aerospace) require reliable models and cannot follow the explosion of AI usage seen in other sectors. Addressing this issue has become a central goal for the further deployment of AI in these industries. This led to a surge of research on uncertainty. However, each research team has attempted to apply their previous expertise to answer this crucial question.

As a result, the field of uncertainty in LLMs is currently very scattered.

The aim of my research internship with the MILES team, under the co-supervision of Yann Chevaleyre and Jamal Atif, was to attempt to unify these diverse research directions. In doing so, we conducted an extensive literature review and discovered a new way to differentiate current methods and measures. This allowed us to reveal that there are **orthogonal directions in research that can be combined, whereas they had previously been competing with each other**. Yet, many challenges still lie ahead. I hope that you will enjoy navigating this blurry world of uncertainty as much as I did.

Happy reading!

2 Acknowledgements

I would like to thank my two supervisors, Yann Chevaleyre and Jamal Atif, as well as my referent teacher, Gabriel Peyré, who suggested I contact this research lab and agreed to be my referent teacher. I am especially grateful for their guidance in helping me improve scientifically.

I would also like to express my special thanks to Julien Remy, a bachelor's student from Columbia University, with whom I had the opportunity to engage in many discussions. He provided valuable assistance with the experiments, and I had the joy of sharing my passion for this topic with him.

I extend my thanks to the entire MILES team: Mathis, with whom I spent a year at the MVA, as well as Augustin, Alex, Garry, Matteo, and Mehdi, my fellow co-interns. I wish for all of us to continue nurturing our shared passion for wolves.

Finally, I would like to acknowledge the PhD students Blaise, Erwan, and Florient, the postdoc Paul, and the newly graduated Dr. Alexandre Verine. Thank you all for your support and inspiration.

Contents

3 Introduction

Critical industries such as healthcare, defense, law, and aerospace require models that are not only powerful but also reliable. These sectors face unique challenges in adopting AI technologies due to the high stakes involved. Mistakes in these fields could lead to catastrophic outcomes, from legal repercussions to potential loss of life. As a result, they have struggled to match the rapid AI expansion seen in less regulated sectors. Despite the clear benefits AI offers, these industries limit their adoption because reliability is paramount. Large language models (LLMs) are particularly relevant, given their remarkable capabilities in processing and generating human-like text. However, their reliability—especially in critical contexts—has raised concerns, making it essential to ensure that LLMs can be trusted in these environments.

Addressing uncertainty in LLMs is key to improving this reliability. Unlike traditional machine learning models, LLMs operate in a probabilistic manner when generating language, making their outputs inherently uncertain. This uncertainty raises concerns about their use in high-stakes industries. For example, a healthcare professional relying on an LLM for diagnosis or a lawyer using it for legal documents could face severe consequences if incorrect or misleading information is provided.

The increasing demand for reliable LLMs has led to a surge in research addressing uncertainty in these models, but this research is highly fragmented. Different teams approach the problem from varying perspectives, resulting in a diverse set of methods, tools, and definitions of uncertainty. This scattering of approaches makes it challenging to form a unified understanding of uncertainty in LLMs.

My research internship with the MILES team, under the supervision of Yann Chevaleyere and Jamal Atif, aimed to unify these disparate research directions. The goal was to identify complementary approaches that could work together, rather than being seen as competing methodologies.

A key distinction that emerged from this research was between prompt-based and logit-based approaches to uncertainty estimation. Prompt-based methods typically manipulate the input provided to an LLM to elicit more reliable or nuanced outputs, while logit-based methods focus on the probabilities assigned to different tokens during the language generation process. Historically, these approaches have been viewed as mutually exclusive or even competitive. However, our research demonstrated that these methods are orthogonal, meaning they can complement each other when combined.

3.1 Why Assessing Uncertainty in LLMs Matters

Understanding the value of assessing uncertainty in LLMs is critical, particularly when models face the phenomenon of hallucination. Hallucinations occur when the model generates factually incorrect or nonsensical outputs but presents them with high confidence. These issues became evident shortly after the public release of models like ChatGPT and remain a significant concern, especially in high-stakes environments such as healthcare and law. A notable case involved a lawyer who had to apologize for submitting fake court citations generated by ChatGPT during legal proceedings [21]. This incident underscores the danger of unchecked confidence in LLM outputs. If the model had expressed uncertainty or admitted it did not know the correct information, such situations could have been avoided.

Beyond mitigating hallucinations, recognizing uncertainty in LLMs allows for better decision-making in collaborative settings. For example, when a model anticipates it might fail or produce unreliable output, it can defer the task to a more advanced model or even a human. This principle of deferral is explored in research like language model cascades, where models of increasing complexity handle tasks based on uncertainty thresholds [11]. Similarly, healthcare systems are being developed where LLMs collaborate with human experts, deferring decisions when the model's confidence is low [28]. This form of human-AI collaboration could significantly improve the reliability of AI in critical applications by ensuring uncertain tasks are managed by the most appropriate party.

3.2 How Does Research Differentiate Uncertainty in LLMs?

What is the capacity that we want to have ? To illustrate how LLMs handle uncertainty, consider a scenario where a user asks, "What is the capital of the UK?" The LLM has two possible responses: it can confidently state "London" or, in a more cautious approach, it can express uncertainty by saying,

"I do not know." This choice reflects the core behavior of LLMs in dealing with uncertainty. In this case, the answer is well-known, so a high degree of confidence is expected. However, in more complex or ambiguous situations, the model's ability to acknowledge uncertainty becomes crucial.

Saying "I do not know," represents a binary approach to uncertainty. Yet, in many cases, a more nuanced uncertainty measure is preferable. Rather than a binary response, the model might provide a probability estimate indicating its confidence. For example, it could indicate that it is 80% certain that its answer is correct. This introduces the concept of *calibration*.

Calibration refers to the alignment between the model's predicted confidence and its empirical accuracy. In a well-calibrated model, if it predicts with 80% confidence, it should be correct 80% of the time in similar cases. Such calibration enables users to better assess the reliability of the model's responses.

Thus, it is possible to understand that uncertainty measures can be differentiated. On one side, binary measures indicate whether the model knows or does not know the answer. On the other side, continuous measures provide probabilistic estimates of correctness. Both approaches have their advantages, depending on the context and the desired level of granularity.

Similarly, there are different methods for assessing uncertainty in LLMs. One common distinction is between *white-box* and *black-box* methods. In white-box approaches, we have access to the model's internal parameters, including its weights and architecture, which allows for more in-depth uncertainty analysis. In contrast, black-box methods work without access to internal workings, relying only on model outputs. These methods are often used with proprietary models but may be limited in how they assess uncertainty.

In our research, we propose a new way of organizing uncertainty assessment methods by introducing the distinction between *prompt-based* and *logit-based* approaches. Prompt-based methods focus on manipulating the input to improve the model's output reliability, while logit-based methods analyze the token-level probabilities during language generation. Traditionally seen as competing methods, we show that they can be complementary. This novel distinction, between prompt-based and logit-based methods, will be explored further in this thesis, offering a new way to understand uncertainty in LLMs.

3.3 Contribution

To develop this new framework in the field of uncertainty assessment in LLMs, we conducted a comprehensive and critical literature review. Throughout this review, we identified several gaps and inconsistencies in the current research. A major finding was that many approaches to uncertainty estimation were treated as competing, whereas they could actually be complementary.

In response, we proposed a new framework for classifying uncertainty assessment methods in LLMs. Specifically, we emphasized the distinction between prompt-based and purely logit-based approaches, offering a unifying structure for previously fragmented research efforts. By combining these methods, we provide a more holistic perspective on uncertainty in LLMs. This led us to develop a **new algorithm that integrates state-of-the-art research 3**.

4 Estimating uncertainty from output logits

4.1 Usage of Entropy

4.1.1 Why Are LLMs Probabilistic?

Large Language Models (LLMs) are fundamentally probabilistic because they generate language by predicting the likelihood of a sequence of tokens. To understand this, we must first break down what tokens and prompts are in the context of an LLM. A *token* is the smallest unit of text that the model processes, which could be a word, part of a word, or even a punctuation mark, depending on the tokenization strategy. A *prompt* is simply the input text provided to the model, based on which the model predicts the next token in the sequence.

LLMs, such as GPT [2], LLAMA [12], Vicuna [30], and Mistral [13], work by predicting the next token in a sequence given a previous context or prompt. Formally, if we denote the prompt by x, the

LLM estimates the probability distribution $P(\cdot|x)$, which represents the likelihood of different tokens being the next token in the sequence. This probabilistic approach ensures that the model doesn't always output the same token for a given prompt, allowing for diversity and creativity in language generation.

Even though the model predicts one token at a time, entire sentences or paragraphs are generated by applying this process iteratively. This means that after predicting the next token, the model updates the prompt by appending this token and then repeats the process to predict the following token. This iterative process of predicting one token at a time makes the model *autoregressive*, as each prediction depends on the previous ones. GPT, for instance, is an autoregressive model that excels at this form of language generation. In contrast, models like BERT [5] are not autoregressive, as they rely on predicting missing words in a sentence rather than generating tokens sequentially.

An LLM's output is driven by the logits, which are the unnormalized scores from the final layer of the model. These logits represent the raw scores associated with each token in the model's vocabulary. The logits are then transformed into probabilities using the softmax function, which ensures that the scores sum up to 1, forming a probability distribution over all possible tokens. The probability $P(t_i|x)$ of a token t_i given the prompt x is computed as:

$$P(t_i|x) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

where z_i represents the logit for token t_i.

In this process, a key concept is the *temperature* parameter. The temperature T scales the logits before applying the softmax function, which impacts the model's tendency to favor certain tokens. The modified probability with temperature is given by:

$$P(t_i|x, T) = \frac{e^{z_i/T}}{\sum_j e^{z_j/T}}$$

The temperature T affects the distribution in the following way:

- When $T = 1$, the model behaves normally without any scaling, using the original probabilities derived from the logits.
- When $T < 1$ (low temperature), the model becomes more deterministic, as the probabilities of the most likely tokens increase, leading to more predictable and less diverse outputs. The model tends to pick the highest-probability token.
- When $T > 1$ (high temperature), the probabilities of less likely tokens are increased, introducing more randomness and allowing for greater diversity in the output. This can lead to more creative or unexpected results, but with the risk of generating incoherent or less relevant text.

Importantly, we do not always want the model to take the most probable token at each step because this "greedy" approach might lead to a locally optimal but globally suboptimal sentence. By allowing for some randomness in token selection through adjusting the temperature, the model can sometimes generate more coherent and natural sentences, even if it means taking a less probable token at a particular step. This flexibility in token selection is one of the reasons why LLMs are capable of generating human-like text that is not overly repetitive or rigid.

4.1.2 Entropy: Measure of Uncertainty in Prediction

Once we have a probability distribution over the next token in an LLM's output, a wide range of techniques can be applied to quantify uncertainty and improve model performance. Various papers explore these approaches, such as cascading models, uncertainty regulation, and entropy-based uncertainty quantification. In our work, we focus on entropy because it is a core concept for measuring uncertainty, and it is frequently used in numerous methods for uncertainty estimation in language models.

Definition 1

Entropy:

Entropy is a fundamental concept in information theory, widely used to quantify the uncertainty of a prediction. The entropy $H(X)$ of a random variable X measures the unpredictability of its outcomes. For a discrete random variable X that can take values $\{x_1, x_2, \ldots, x_n\}$ with corresponding probabilities $\{p(x_1), p(x_2), \ldots, p(x_n)\}$, the entropy is defined as:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log p(x_i)$$

Here, $p(x_i)$ represents the probability of outcome x_i, and the logarithm is typically taken in base 2, which expresses the entropy in bits. The entropy quantifies the average amount of information needed to describe the outcome of X. In the context of LLMs, entropy is used to measure the uncertainty in the model's prediction of the next token. High entropy suggests that the model is uncertain about the next token, whereas low entropy indicates that the model is more confident in its prediction.

Entropy and Thresholds in LLM Predictions When you have your measure of entropy one possibility is that if it exceeds a certain threshold, the model can respond with "I do not know," indicating that it is unsure of the correct answer. This approach helps manage cases where the model would otherwise generate hallucinations or incorrect outputs with high confidence.

AUROC: Evaluating Entropy as an Uncertainty Estimator To evaluate how well entropy functions as an uncertainty estimator, we can assess its performance on a question-answer (QA) dataset. For each model prediction, we have a binary label indicating whether the answer was correct or incorrect. The effectiveness of entropy in distinguishing between these correct and incorrect predictions can be evaluated using the Area Under the Receiver Operating Characteristic (AUROC) curve.

The AUROC is a widely-used metric for binary classification tasks, and it provides insight into how well a model's uncertainty measure (in this case, entropy) can separate true positives (correct predictions) from false positives (incorrect predictions). To understand the AUROC curve, we must define several terms:

- **True Positives (TP)**: The cases where the model correctly predicted the label (i.e., the model was confident, and the prediction was correct).
- **False Positives (FP)**: The cases where the model was confident, but the prediction was incorrect.
- **True Negatives (TN)**: The cases where the model correctly identified that it should be uncertain (i.e., the entropy was high, and the model abstained from answering).
- **False Negatives (FN)**: The cases where the model was uncertain, but the prediction was actually correct.

Definition 2

Area Under the Receiver Operating Characteristic (AUROC):
The AUROC measures the model's ability to distinguish between correct and incorrect predictions based on its confidence (entropy). In our case, it plots the true positive rate (TPR) against the false positive rate (FPR) at different thresholds of entropy. The true positive rate is defined as:

$$TPR = \frac{TP}{TP + FN}$$

and the false positive rate is:

$$FPR = \frac{FP}{FP + TN}$$

A higher AUROC indicates that entropy is a good estimator of uncertainty—i.e., when entropy is high, the model is more likely to be incorrect, and when entropy is low, the model is more likely to be correct. This enables to measure the quality of the entropy as a measure for the task independently of the threshold used to force the model to say I do not know.

Variable Length Generations Natural language sentences vary in length. On average, longer sequences tend to have lower joint likelihoods due to the conditional independence of token probabilities. As the length of a sequence N increases, the joint likelihood diminishes exponentially with N . Consequently, the negative log-probability grows linearly with N, meaning that longer sentences typically contribute more to overall entropy [18]. Normalizing log-probabilities by sentence length when estimating entropy essentially assumes that the expected uncertainty of the generated text is independent of sentence length. While this assumption is sometimes approximately valid, in other cases, longer sentences may inherently carry more uncertainty [20].

4.1.3 Recent Interpretability Research on Neurons Responsible for Changing the Entropy of the Distribution

Recent advancements in explainability research have identified specific neurons in Large Language Models (LLMs) that play a key role in regulating entropy. These neurons, termed "entropy neurons," are responsible for flattening the probability distribution over the next token, thereby increasing uncertainty in the model's prediction. Flattening the distribution leads to a more even spread of probabilities across tokens, which reflects higher uncertainty and allows the model to manage cases where the next token is not obvious.

One of the notable findings from this research is that certain neurons project into the null embedding space, despite the model being trained with weight decay, a regularization technique intended to prevent large weights 9. Interestingly, these neurons maintain a non-zero norm even in the null space. This suggests that in order to make robust predictions, the model must intentionally flatten the distribution at times to capture more uncertainty. The presence of neurons projecting into the null space implies that controlling the flatness of the distribution is a fundamental part of the model's operation, especially when handling uncertainty.

MLP Layers and Neurons in Transformers Central to understanding the role of entropy neurons is the structure of the Multi-Layer Perceptron (MLP) layers within the transformer architecture. Given a normalized residual stream hidden state $x \in \mathbb{R}^{d_{model}}$, the output of an MLP layer is defined as:

$$MLP(x) = W_{out}\sigma(W_{in}x + \beta_{in}) + \beta_{out}$$

Where:

- $W_{in} \in \mathbb{R}^{d_{mlp} \times d_{model}}$ and $W_{out} \in \mathbb{R}^{d_{model} \times d_{mlp}}$ are the learned weight matrices.

- β_{in} and β_{out} are the learned biases.
- σ is an element-wise nonlinear activation function, typically the GeLU activation function [**?**].

In this context, a *neuron* refers to an entry in the hidden layer of the MLP. We use n to denote the activation values of these neurons, which are the output of the activation function σ. Additionally, $w_{out}^{(i)} \in \mathbb{R}^{d_{model}}$ represents the output weights of neuron i, i.e., the i-th column of the matrix W_{out}.

Replication of the Key-Query Matrix Analysis In my own replication of these findings, I focused on examining the key-query matrix of the transformer model, particularly investigating the rank of the product of the key-query matrix and the resulting eigenvalues. The original study found that certain layers exhibited a decay in the influence of the key-query matrix, which allowed for the identification of the layers where the entropy neurons had the most significant effect.

In my analysis, I observed that if I took a random vector and projected it onto each of the eigenvectors of the key-query matrix, the projection was predominantly onto the layers responsible for the zero eigenvalues. This result implies that noise (i.e., random vectors) naturally projects onto the zero eigenvalue components of the matrix, which corresponds to the layers where entropy neurons are most active. These findings suggest that in layers with a higher concentration of zero eigenvalues, the model is more likely to flatten the token distribution, introducing higher uncertainty.

This behavior aligns with the hypothesis that entropy neurons are central to managing uncertainty within the model. Their ability to manipulate the token distribution by projecting into the null space allows the model to balance between confident predictions and uncertainty, thus improving its overall reliability in tasks involving probabilistic outputs.

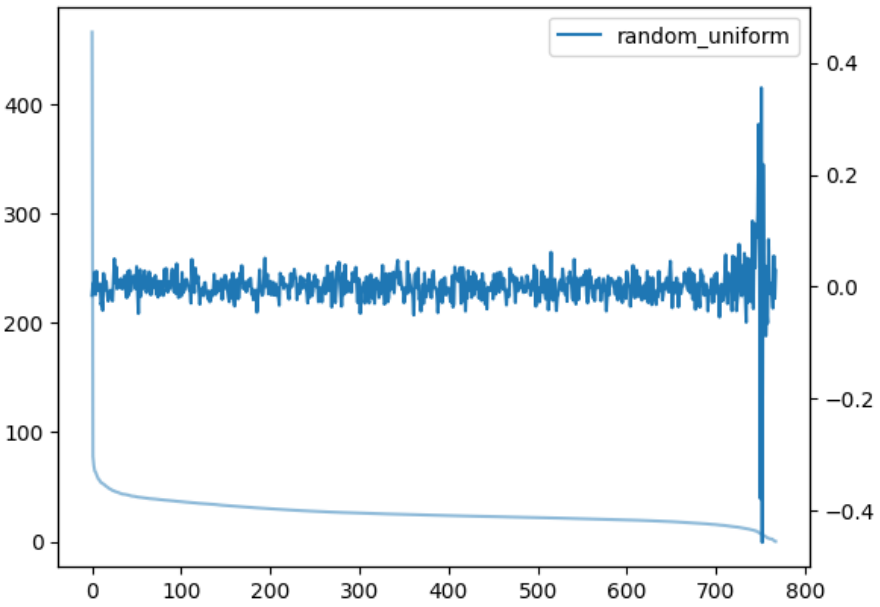


Figure 1: Decay of the eigenvalues of the key-query (light blue line, going from 450 to 0) and projection of a random vector on each of the corresponding eigenvectors (with values ranging from -0.5 to 0.5). It can be observed that the uniform vector is projecting mostly to the null space. Thus, the null space is responsible for playing with noisy input.

4.2 Clustering Answers to Improve Uncertainty Estimation

While entropy is a powerful tool for measuring uncertainty, relying solely on it can be insufficient. Natural language is inherently diverse, with multiple ways to express the same idea. For example, the responses "Paris" and "It's Paris" are semantically equivalent answers to the question "What is the capital of France?" However, if we merely analyze the entropy of the distribution over these answers without considering their meaning, we might treat them as distinct answers, which would artificially

inflate the entropy. This is problematic because the model is not demonstrating a lack of capacity—it is merely expressing the same correct answer in different ways.

Semantic meaning plays a critical role in ensuring trustworthiness. A system can remain reliable even when it uses different phrasings to convey the same message. However, if a system produces inconsistent or contradictory meanings, it undermines reliability. Therefore, to improve uncertainty estimation, we propose clustering semantically equivalent answers. This allows us to focus on meaningful variations in the model's predictions rather than superficial differences in expression.

The Impact of Clustering on Entropy By clustering responses that have the same meaning, we can refine the estimation of entropy. Due to the sublinear nature of the logarithm in the entropy formula, combining the probabilities of semantically equivalent answers results in a lower entropy value than when treating these answers as separate entities. Let's consider two scenarios to illustrate this:

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 s	Likelihood $p(s \mid x)$	Semantic likelihood $\sum_{s \in C} p(s \mid x)$
Paris	0.5	0.5
Rome	0.4	0.4
London	0.1	0.1
Entropy	0.94	

(b) Scenario 2: Some semantic equivalence

Answer s	Likelihood $p(s \mid x)$	Semantic likelihood $\sum_{s \in C} p(s \mid x)$
Paris	0.5	0.9
It's Paris	0.4	
London	0.1	0.1
Entropy	0.94	0.33

In the first scenario, all generated answers are distinct and therefore have no semantic overlap, resulting in no change to the entropy when considering semantic equivalence. The entropy of the original distribution is identical to the semantic entropy.

In the second scenario, however, some of the answers are semantically equivalent (e.g., "Paris" and "It's Paris"). By grouping these equivalent answers together and summing their probabilities, the entropy drops significantly from 0.94 to 0.33, better reflecting the actual uncertainty in the model's responses. This shows how semantic clustering can improve the accuracy of uncertainty estimation by reducing redundancy in the model's output.

Clustering Semantically Equivalent Sentences The remaining challenge is how to effectively cluster semantically equivalent sentences. To do this, we can use a natural language model trained specifically for detecting whether two sentences imply each other, known as a natural language inference (NLI) model. The idea is to leverage this model to compare sentence pairs. Specifically, for each generated sentence, we construct two inputs:

- The first input is the concatenation of the question and one generated answer.
- The second input is the concatenation of the question and another generated answer.

The NLI model is then asked whether the first input entails the second. To ensure robustness, the process is reversed: we also ask whether the second input entails the first. If both entailment scores exceed a predefined threshold, we consider the two answers to be semantically equivalent.

This process effectively clusters semantically similar answers, allowing us to sum their probabilities and recalculate the entropy based on meaning, not just surface-level token differences.

This yields therefore the following algorithm :

Algorithm 1 Bidirectional Entailment Clustering from [**?**]

Require: context x, set of seqs. $\{s^{(2)}, \ldots, s^{(M)}\}$, NLI classifier $\mathcal{M}$, set of meanings $\mathcal{C} = \{\{s^{(1)}\}\}$
  for $2 \leq m \leq M$ **do**
    for $c \in \mathcal{C}$ **do**
      $s^{(c)} \leftarrow c_0$ ▷ Compare to already-processed meanings.
      $left \leftarrow \mathcal{M}(cat(x, s^{(c)}, "\langle g/\rangle", x, s^{(m)}))$ ▷ Use first sequence for each semantic-class.
      $right \leftarrow \mathcal{M}(cat(x, s^{(m)}, "\langle g/\rangle", x, s^{(c)}))$ ▷ Does old sequence entail new one?
      if $left$ is entailment **and** $right$ is entailment **then** ▷ Vice versa?
        $c \leftarrow c \cup \{s^{(m)}\}$ ▷ Put into existing class.
      end if
    end for
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{s^{(m)}\}$ ▷ Semantically distinct, gets own class.
  end for
  return $\mathcal{C}$

Summary of the Algorithm This approach, from [18], can be summarized in the following 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 that have equivalent meanings using a bi-directional entailment algorithm based on natural language inference.

3. **Entropy Estimation**: Approximate semantic entropy by summing the probabilities of sentences that share the same meaning (as shown in Table 1), and then compute the resulting entropy.

By following these steps, we can more accurately estimate uncertainty by accounting for the semantic equivalence of answers, rather than just their surface-level forms.

In [18], the authors show that measuring entropy with semantic clustering yields better results than using plain entropy, and this is independent of the model size 10. Once again to measure the performances of this measure, they used the AUROC on a certain type of Question Answering Dataset, in that case, the TriviaQA dataset [14].

4.3 Seeing the answers as a graph with edges being their semantic link

In the previous section, [18] proposed grouping semantically equivalent sentences. However, NLI models provide much richer information as they relate each sentence directionally through entailment, contradiction, or neutral relations. The following section aims to group together approaches that have emerged from the semantic clustering idea, either using alternative clustering methods based on graphs, such as spectral clustering, or directly deriving a graph kernel from the graph itself, which can then be used in a new entropy measure, the Von Neumann Entropy.

Before diving into these approaches, let us first present a visual representation of this concept, as shown in 2.

4.3.1 Using Spectral Clustering on the Graph of Answers

The motivation behind [19] was to extend the semantic entropy technique to black-box models, where access to internal model representations, such as logits, is unavailable.

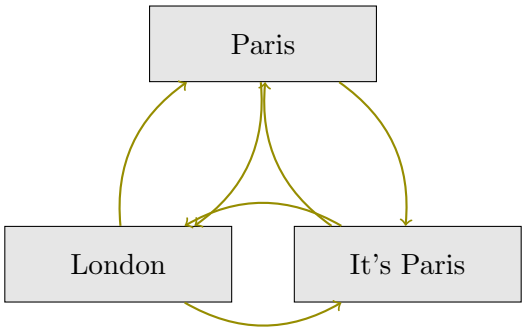


Figure 2: This graph shows the relationship between three possible answers: "Paris", "It's Paris", and "London" to the question, "What is the capital of France?". The arrows represent the relationships between the answers based on Natural Language Inference (NLI) weights, with each edge representing the probability of contradiction, entailment, or neutral stance.

A significant challenge in this task arises from the fact that semantic relationships between sentences are not always clear-cut. For instance, when posed with the question, "What city was Zeus the patron god of?", three possible answers—"Olympia," "Zeus was the patron god of Olympia, Greece," and "Corinth"—could be grouped into two semantic clusters, with the first two responses being more closely related. However, deciding whether two responses have the same meaning is often not a simple binary decision. In the Zeus example, responses like "Olympia" and "Greece" are not completely identical but neither are they entirely different in meaning.

Additionally, the semantic equivalence judged by a Natural Language Inference (NLI) model—or any similar measure—is not guaranteed to be transitive. This means that rigid clustering of responses can sometimes oversimplify the true relationships between answers. Therefore, a more nuanced approach that allows for a continuous and flexible interpretation of semantic similarity is preferable.

Spectral clustering offers a solution by treating the set of responses as a graph, where each response is represented by a node and the edges between nodes reflect semantic similarity scores between pairs of responses. By applying spectral clustering to this graph, we can group the responses into clusters that reflect more nuanced, graded relationships, rather than relying on strict binary classifications of whether two answers are equivalent. This provides a more accurate representation of the uncertainty present in the set of responses.

Sum of Eigenvalues of the Graph Laplacian Let the responses be denoted by s^{j_1} and s^{j_2}. In [**?**], the authors did not have access to the logits because they focused on black-box models. Instead, they worked with pairwise similarities a_{j_1,j_2} between response s^{j_1} and s^{j_2}.

Definition 3

Symmetric Normalized Graph Laplacian:
Given a fixed input x, each generated response is treated as a node in a graph, and the symmetric weighted adjacency matrix is defined as:

$$W = (w_{j_1,j_2})_{j_1,j_2=1,\ldots,m} \quad \text{where} \quad w_{j_1,j_2} = \frac{a_{j_1,j_2} + a_{j_2,j_1}}{2}.$$

The symmetric normalized graph Laplacian is then computed as:

$$L = I - D^{-1/2} W D^{-1/2},$$

where D is the degree matrix, defined as:

$$D_{j_1,j_2} = \begin{cases} \sum_{j' \in [m]} w_{j_1,j'} & \text{if } j_1 = j_2, \\ 0 & \text{if } j_1 \neq j_2. \end{cases}$$

A continuous measure of the number of semantic clusters can be defined using the eigenvalues $\lambda_1 < \cdots < \lambda_m$ of the Laplacian L as a measure of uncertainty:

$$U_{\text{EigV}} = \sum_{k=1}^{m} \max(0, 1 - \lambda_k).$$

This measure relates the uncertainty in the responses to the eigenvalues of the Laplacian matrix, which provides insight into the semantic clusters of the responses.

Connection Between Eigenvalues and Clusters The relationship between the number of semantically equivalent sets of sentences and the eigenvalues of the Laplacian is captured by the following theorem:

Theorem 3

Connected components and Laplacian[31]:
The multiplicity of the eigenvalue 0 of L is equal to the number of connected components in the graph represented by W.
In other words, if W is a binary matrix (indicating that two responses are either connected or not), the number of zero eigenvalues corresponds to the number of distinct semantic clusters. In the case of a continuous W, the graph typically has one connected component, but spectral clustering leverages the distribution of the eigenvalues to determine the number of clusters.

This approach provides a more continuous and flexible measure of uncertainty compared to traditional binary clustering methods. As illustrated in figure 11, depending on the type of question—whether it elicits more or less diverse answers—the behavior of the decay of the eigenvalues of the semantic graph will vary.

Yet, an even more precise method exists.

4.3.2 Using Graph Kernels

In [22], the authors developed a method to compute semantic uncertainty that accounts for subtle similarities between text generations, providing a more accurate uncertainty quantification.

A critical limitation of Semantic Equivalence (SE), as highlighted in [22], is that it captures semantic relations between generated texts solely through equivalence relations. This does not provide a distance metric in the semantic space, which would allow for capturing more nuanced semantic similarities between generations. For instance, SE treats "apple" as equally distinct from "house" as it does from "granny smith," even though the latter pair is much more closely related.

In contrast to the previous subsection, where graph clustering was used, [22] proposed using graph kernels for a more refined approach.

Before diving into this concept, let's first recall what a kernel is.

Definition 4

Positive Semi Definite Kernel:
For a set $X \neq \emptyset$, a symmetric function $K : X \times X \to \mathbb{R}$ is called a PSD kernel if for all $n > 0$, $x_i \in X$, $\alpha_i \in \mathbb{R}$

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j K(x_i, x_j) \geq 0.$$

For a finite set X, a PSD kernel is a PSD matrix of the size $|X|$.

Next, it is possible to define semantic kernels, denoted K_{sem}, as unit trace[1] positive semidefinite

kernels over the finite domain of generated texts. Unit trace PSD matrices are also called density matrices. These kernels should, informally speaking, capture the semantic similarity between the texts such that $K(s_1, t_1) > K(s_2, t_2)$ if and only if texts s_1 and t_1 are more semantically related than texts s_2 and t_2. Analogously, we define semantic kernels over semantic clusters of texts, in which case the kernel should capture the semantic similarity between the clusters.

Graph Kernels When a semantic graph is obtained, just as it has been given in the previous subsection, KLE calculates graph kernels over semantic graph nodes to compute a distance measure. Since graphs are discrete and finite, any positive semidefinite matrix would be a kernel over the graph. However, we seek kernels that exploit knowledge about the graph structure. We therefore adopt Partial Differential Equation (PDE) and Stochastic Partial Differential Equation (SPDE) approaches to graph kernels [16]. If $u \in \mathbb{R}^n$ is a signal over the nodes of a graph, the heat kernel is a solution to the partial differential equation $\partial u/\partial t + Lu = 0$ and the Matérn kernel is a solution to the stochastic differential equation,

$$\left(\frac{2\nu}{\kappa^2} + L\right)^{\frac{\nu}{2}} u = w,$$

where w is white noise over the graph nodes and L is the graph Laplacian defined above. The corresponding solutions to these equations are:

$$K_t = e^{-tL} \quad \text{[HEAT]}, \quad K_{\nu\kappa} = \left(\frac{2\nu}{\kappa^2} I + L\right)^{-\nu} \quad \text{[MATÉRN]}.$$

These kernels allow for the incorporation of a distance measure that reflects the graph's locality properties

Kernel Combination KLE offers the additional flexibility of combining kernels from various methods (e.g., multiple NLI models, different graph kernels, or other methods). For example, we can combine multiple kernels using convex combinations,

$$K = \sum_{i=1}^{P} \alpha_i K_i,$$

where $\sum_{i=1}^{P} \alpha_i = 1$.

The von Neumann Entropy We propose to use the von Neumann entropy (VNE) to evaluate the uncertainty associated with a semantic kernel.

Definition 5

Von Neumann Entropy:
For a unit trace positive semidefinite matrix $A \in \mathbb{R}^{n \times n}$, the von Neumann entropy (VNE) is defined as

$$\text{VNE}(A) = -\text{Tr}[A \log A].$$

It can be shown that $\text{VNE}(A) = \sum_{i=1}^{n} -\lambda_i \log \lambda_i$ where λ_i, $1 \leq i \leq n$ are the eigenvalues of A. Within this definition, we assume $0 \log 0 = 0$. This reformulation shows that VNE is, in fact, the Shannon entropy over the eigenvalues of a kernel. From that it is possible to define Kernel Language Entropy.

Definition 6

Kernel Language Entropy:
Given a set of LLM generations $S_1, \ldots, S_N$, an input x, and semantic kernel K_{sem} over these generations and input, we define Kernel Language Entropy (KLE) as the von Neumann entropy of a semantic kernel K_{sem}:

$$\text{KLE}(x) = \text{VNE}(K_{\text{sem}}).$$

5 Estimating uncertainty by prompting LLM multiple times

In the previous section, the usage and semantic clustering of the model's output were developed to the fullest extent possible. Indeed, a lot of uncertainty measures can be also achieved by just paying correctly with the prompt [17].

However, this approach is sometimes insufficient. Research on large language models (LLMs) has often focused on reformulating questions in a certain way, particularly simplifying them, to achieve better performance [**?**]. Since the first paper that aimed to measure LLMs' uncertainty, researchers have been trying to find ways to gauge uncertainty by reformulating prompts correctly.

5.1 Transforming Open Questions to Yes-No

In particular, in the work proposed by [15], *Language Models (Mostly) Know What They Know*, the authors reformulated prompts to measure uncertainty.

The initial prompt in this scenario is the question, "Who was the first President of the United States?" When presented with this prompt, the language model generates 10 potential answers. These answers are then processed further by transforming them into a new format for evaluation.

For one of the answers, specifically, "George Washington," the prompt is reformulated into a yes-no question. The reformulated prompt now asks: "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:"

This transformation allows the model to evaluate the truthfulness of the proposed answer by selecting either true or false. By examining the probability assigned to the "True" token, the model's certainty about the answer can be measured. The probability of the token **True** is then evaluated. This approach yields significant results as the model is able to distinguish between correct and incorrect answers based on the probability of the token **True**, as depicted in Figure 12.

Figure **??** illustrates the distribution of probabilities assigned to the token **True** across various answers generated by the model. The x-axis represents the probability assigned to **True**, while the y-axis shows the fraction of problems for which the model assigned a given probability.

The histogram is divided into two groups: green bars represent correct samples, and blue bars represent incorrect samples. The results show that when the model is more confident in an answer (i.e., assigning a higher probability to the token **True**), the likelihood of the answer being correct is significantly higher. Conversely, answers with lower probabilities assigned to **True** tend to be incorrect. This separation between correct and incorrect answers demonstrates that the model has a measurable sense of its uncertainty regarding the correctness of its outputs. These results highlight that, while imperfect, LLMs can offer meaningful insights into their own confidence levels.

5.2 Selecting Adversary Context

Reformulating prompts for better usage will be a recurring theme in the following sections. However, the authors introduce a novel approach—iterative prompting—where the LLM is queried multiple times with increasingly complex prompts.

In the first step, the authors [36] take the initial prompt and reformulate it into a question-answering format. The goal is then to iteratively incorporate adversary sentences into the it.

Prompt 1 :

Consider the following question Q: *What is the capital of the UK?*
A:

Prompt 2 :

Consider the following question Q: *What is the capital of the UK?* One answer to question Q is Paris. Provide an answer to the following question Q: *What is the capital of the UK?*
A:

Prompt 3 :

Consider the following question Q: *What is the capital of the UK?* One answer to question Q is Paris. Another answer to question Q is Paris. Provide an answer to the following question Q: *What is the capital of the UK?*
A:

The idea is to observe the probability of the correct token—in this case, the token `London`—and track how it varies throughout the adversarial iterative prompting procedure. As shown in 13, the probability of the token `London` is not significantly affected by the repetition of the incorrect token `Paris`. In fact, the probability remains close to 1 most of the time, only dropping slightly after 100 repetitions of the wrong answer. This indicates that the model is not influenced by the adversarial input of `Paris`. The model exhibits low uncertainty regarding the question `What is the capital of the UK?`.

Conversely, consider the question `What is the national instrument of Ireland?`. The model's behavior is completely different, as depicted in 14. Initially, the model assigns a very high probability to the correct answer, `the harp`. However, the probability of this answer dramatically crashes after just one repetition of the wrong answer, `Uilleann pipes`. This indicates that the model is highly influenced by previous answers, even after just one incorrect answer, suggesting that it lacks confidence in this case.

Important note on the limitation of entropy: In the Irish musical instrument example, the initial entropy was very low since the model's prediction of `the harp` was highly confident. Thus, if one had only relied on entropy, they might have been misled into thinking the model was confident, when in fact it was highly susceptible to adversarial inputs.

5.3 Does it replicate?

However, the results from [36] were not shared with accompanying code, and the algorithm used was Gemini Pro [29], an algorithm to which we did not have access. Our motivation, therefore, was to attempt to replicate these results using models for which we had access to the logits.

5.4 Attempting to replicate the adversarial iterative prompting

To replicate the experiment, we used the following type of prompt:

`Consider the following question Q: What is the capital of the UK? One answer to question Q is Paris. Another answer to question Q is Paris. Another answer to question Q is Paris. Provide an answer to the following question: What is the capital of the UK? Answer:.`

We conducted this experiment on different models:

- Bloom 7b [35]
- Llama2 [12]
- Vicuna 7b [30]
- Mistral 7b [13]

This yielded the results depicted in 3, which illustrate the behavior of different models in response to the question `What is the capital of the UK?`. For example, if we observe the yellow lines

representing the probabilities from the Mistral 7b model—where the bold line represents the correct token `London`, and the dashed line represents the incorrect token `Paris`—we can see that, initially, the model answers correctly, with the bold line positioned above the dashed line. This is the case for all models at the start.

As expected, we observe that the probability of the incorrect token increases while the probability of the correct token decreases. At some point, the bold and dashed lines even intersect. However, after this intersection, the behavior becomes much more chaotic compared to the monotonic behavior described by [36], indicating that our results do not follow the same smooth pattern.

The behavior of different models in response to other questions can be seen in 15, 16, and 17.

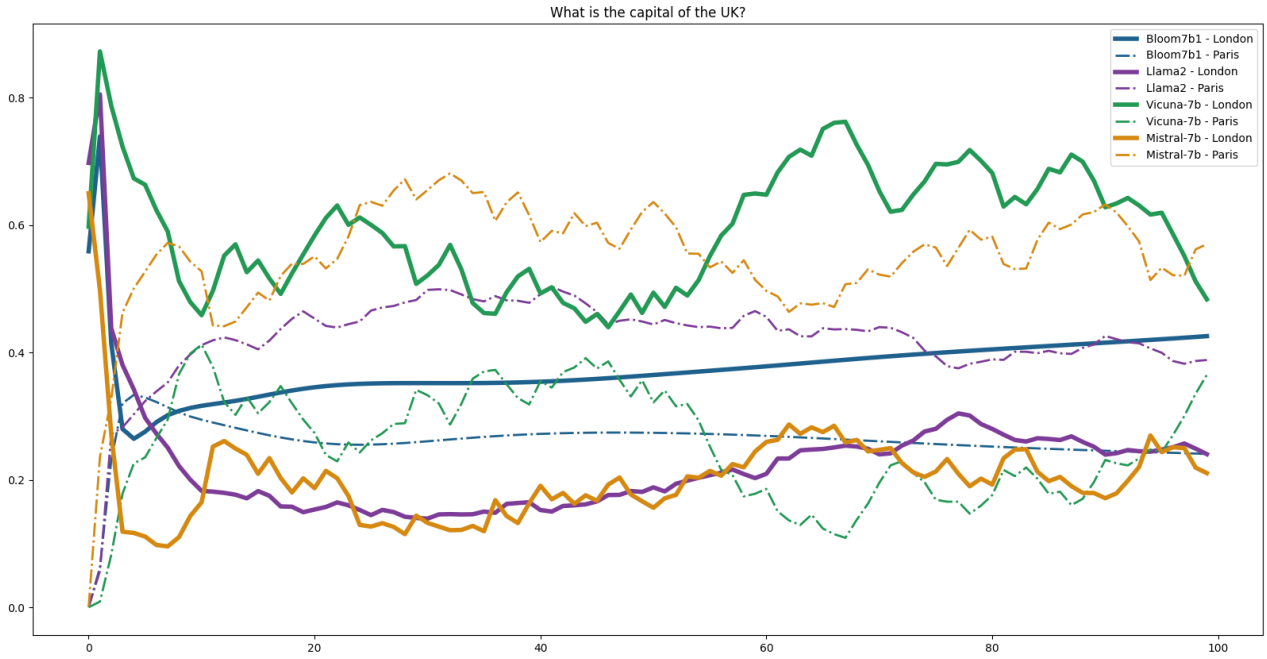


Figure 3: The X-axis represents the repetition of the adversary answer, `Paris`, to the question `What is the capital of the UK`. The Y-axis represent the probability of tokens, either the good answer, `London`, in bold line, or the wrong one, `Paris`, in dashed line. Each color pair corresponds to a different LLM. The prompt used was the same as the one initially proposed by [36].

This motivated us to explore the experiment further by testing a different type of prompt.

5.5 Exploring Another Type of Iterative Prompting

In the previous experiment, we observed an oscillation in the probabilities of the correct and incorrect tokens. Since LLMs are based on the transformer architecture, this led us to hypothesize that these oscillations might be related to the model's attention mechanism, particularly the positional encoding.

Positional encoding assigns a positional score to the tokens in a sentence, often using a sinusoidal function. The oscillations we observed in 3 reminded us of this, but the period did not match. Positional encoding typically operates over sequences of up to 10,000 tokens, whereas the oscillations we observed occurred within 100 repetitions of sentences that were far shorter than 100 tokens.

This observation inspired us to try a new approach: instead of only repeating the adversarial answer, we also repeated the initial correct answer. The idea was to create a balanced prompt with an equal number of correct and incorrect answers repeated throughout.

The objective of this new experiment was to analyze the behavior of the different models under this modified prompting setup, tracking the probabilities of the tokens.

`Consider the following question Q:What is the capital of the UK? Answer: Paris. What is the capital of the UK? Answer: Paris. What is the capital of the UK? Answer: Paris. Provide an answer to the following question: What is the capital of the UK? Answer:`

This yields the following results that can be seen in 4.

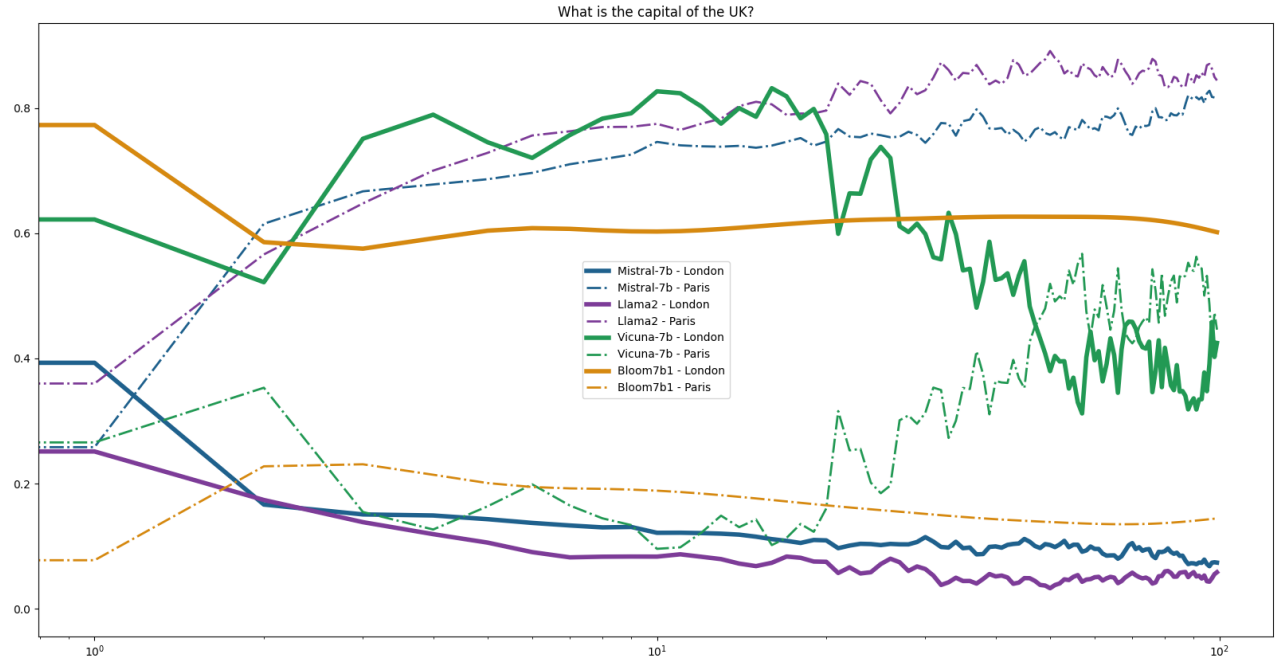


Figure 4: The X-axis represents the repetition of the adversary answer, `Paris`, to the question `What is the capital of the UK`. The Y-axis represent the probability of tokens, either the good answer, `London`, in bold line, or the wrong one, `Paris`, in dashed line. Each color pair corresponds to a different LLM. The prompt used was different from the one initially proposed by [36].

Compared to 3, what can be observed is a more monotonic behavior in the different probabilities. Additional results from this second experiment can be seen in 18, 19, and 20.

5.5.1 Is there an explanation for the observed difference?

In the previous section, we observed that depending on the prompt setup—whether the wrong answer was repeated as many times as the question itself—the behavior of the LLMs differed. The behavior of the LLM is therefore much more sensitive to the prompt structure than what was suggested in [36].

The intuition drawn from these experiments is that it seems the relative proportion of wrong answers to correct ones is what matters most.

Interestingly, [36] also highlights a distinction between in-weight and in-context learning that aligns with this observation. The authors attempt to mathematically model the behavior of the prompt based on whether the question appeared frequently during training or not.

Let $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V \in \mathbb{R}^{d\times d}$ represent the query, key, and value matrices, respectively. A self-attention head with query X and context Z, X is defined as:

$$f(Z; \mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V) = \text{Softmax}\left(\frac{1}{\sqrt{d}} E^\top \mathbf{W}^Q (Z\mathbf{W}^K)^\top\right) Z\mathbf{W}^V$$

where the output of the softmax is a row vector of length n.

If X has appeared frequently in the training data, the parameters $\mathbf{W}^Q$ and $\mathbf{W}^K$ could be learned such that $E^\top \mathbf{W}^K (\mathbf{W}^K)^\top X$ is large, meaning X is within the space spanned by the large principal components of the key-query matrix product. In this case, regardless of the in-context information Z, the probability assigned to X will dominate the softmax, resulting in:

$$\text{Softmax}\left(\frac{1}{\sqrt{d}} E^\top \mathbf{W}^Q (Z\mathbf{W}^K)^\top\right) Z \approx X^\top,$$

and thus $f(Z; \mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V) \approx P(\cdot|X)$.

However, if X has not appeared frequently in the training data and vector Y is repeated multiple

times in Z, $E^\top \mathbf{W}^K (\mathbf{W}^K)^\top X$ could be small, as X may not span the principal components of the key-query matrix product. In this case, the function $f(Z; \mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V)$ would approximate $Y^\top$, and:

$$\text{Softmax}\left(\frac{1}{\sqrt{d}} E^\top \mathbf{W}^Q (Z\mathbf{W}^K)^\top\right) Z \approx Y^\top.$$

However, this explanation relies on the assumption that if the question has been seen many times, it should be in the span of the principal components of the key-query matrix. This assumption is not explicitly tested or supported by existing literature. While it may be true in the case of Principal Component Analysis (PCA)—where data points frequently present in the dataset will lie within the span of the principal components—this is not necessarily the case with the key-query matrix in a transformer model. It is unclear why the same would apply here.

Furthermore, [36] points out that even if X lies in the span of the principal components, repeating Y multiple times in Z would amplify Y's weight in the softmax by a factor of t, where t is the number of repetitions. When t is large enough, this could dominate the weight assigned to X, resulting in Y being the answer.

This could explain the influence of the relative frequency of correct and incorrect answers. However, this explanation suggests a monotonic behavior in the token probabilities. In other words, the probability (or score) of the correct token, `"London"` in our example, should continuously decrease after a certain number of repetitions, while the probability of the incorrect token, `"Paris"`, should increase. Yet, as observed earlier, this was not the case.

5.6 But You Don't Always Have Access to the Adversarial Prompt in the Wild

In the previous examples, estimating confidence required access to the adversarial prompt. While this is possible in controlled experimental setups, it becomes much more challenging when deploying models in real-world scenarios. Therefore, instead of relying on an adversarial prompt, the idea is to use the model's own answers from the first iteration of the prompt to estimate uncertainty.

Let's begin with an example:

1) **Prompt 1**: *"What is the capital of the UK ?"*
LLM: { *"London"* (proba=0.5), *"Madrid"* (0.3), *"Paris"* (0.2) } := **Y1**

2) **Prompt 2**: *"What is the capital of the UK? Knowing that one possible answer is {London, Madrid, Paris}. What is the capital of the UK ?"*
LLM: { *"London"* (proba=0.7), *"Madrid"* (0.2), *"Paris"* (0.1) } := **Y2**

As mentioned earlier, in the adversarial iterative prompting technique deployed by [36], the key is to measure the extent to which the last answer depends on the previous answers.

The two answers can be viewed as random variables, allowing us to compute the mutual information between $\underline{Y_1}$ and $\underline{Y_2}$.

Definition 7

Mutual Information as the Product of Joint and Marginal Probabilities:
The mutual information $I(Y_1; Y_2)$ between two random variables Y_1 and Y_2 quantifies the amount of information obtained about one random variable through the other. It can be defined as:

$$I(Y_1; Y_2) = \sum_{y_1, y_2} p(y_1, y_2) \log \frac{p(y_1, y_2)}{p(y_1)p(y_2)}$$

where $p(y_1, y_2)$ is the joint probability distribution of Y_1 and Y_2, and $p(y_1)$ and $p(y_2)$ are the marginal distributions of Y_1 and Y_2, respectively. This definition emphasizes the ratio between the joint probability and the product of the marginal probabilities.

Definition 8

Mutual Information as the Difference between Entropy and Conditional Entropy: Alternatively, the mutual information between Y_1 and Y_2 can be expressed in terms of entropy. It represents the reduction in uncertainty of Y_1 after observing Y_2 and is defined as:

$$I(Y_1; Y_2) = H(Y_1) - H(Y_1|Y_2)$$

where $H(Y_1)$ is the entropy of Y_1 and $H(Y_1|Y_2)$ is the conditional entropy of Y_1 given Y_2. This definition shows how much knowing Y_2 reduces the uncertainty in Y_1.

Just as with entropy in the previous section, mutual information can also be used as a measure of uncertainty. The idea is to select a threshold, such that if the mutual information between the answers exceeds this threshold, the model should respond with `I do not know`.

In the following subsection, we will introduce the formal framework used to compute the mutual information as described above.

5.7 Formalizing the dependence between answers for iterative prompting

Let $\mathcal{X}$ be the space of finite text sequences, that is $\mathcal{X} \subset \Sigma^*$ where Σ is a finite alphabet (and $\Sigma^* = \bigcup_{n=1}^{\infty} \Sigma^n$). Moreover, consider a family of conditional distributions $\mathcal{P} = \{\mu : \mathcal{X} \to [0,1] \mid \sum_{x \in \mathcal{X}} \mu(x \mid x') = 1 \ \forall x' \in \mathcal{X}\}$. In the following, we let $P \in \mathcal{P}$ be the ground-truth conditional probability distribution over text sequences (responses) given a prompt, and we let $Q \in \mathcal{P}$ be the learned language model. Given a fixed query $x \in \mathcal{X}$ and possible responses $Y_1, \ldots, Y_t$, we define a family of prompts $\mathcal{F} = \{F_t : \mathcal{X} \to \mathcal{X} \mid t \in \mathbb{N}\}$, such that $F_t(x, Y_1, \ldots, Y_t)$ is defined as:

Consider the following question: Q: x
One answer to question Q is Y_1. Another answer to question Q is Y_2. ...
Another answer to question Q is Y_t.
Provide an answer to the following question:
Q: x. A:

Let μ, μ' be distributions supported on set $\mathcal{Z} = \mathcal{Z}_1 \times \cdots \times \mathcal{Z}_n$ where $(\mathcal{Z}_i)_i$ is a collection of countable sets. In particular, the idea is to use the likelihood of the model, as explained in the introduction, as the distribution over the $\mathcal{X}$.

To compute the mutual information, using the definition of the Mutual Information as the Product of Joint and Marginal Probabilities, one need to do the following steps :

What can be observed is that, in order to compute the joint distribution μ, it is necessary to apply Bayes' formula: $p(A \cap B) = p(A|B)p(B)$, where A and B are two events, and p is a probability measure. Moreover, since we only have access to the conditional probabilities for the second set of answers, we need to marginalize over the first set of answers. This is done by summing over all the possible previous answers.

Combinatorial Problems It quickly becomes clear that this marginalization leads to a significant combinatorial challenge. One way to address this is by sampling answers from the models only once.

Thus, [36] proposed a method to combine answers. The authors clustered the answers using an F1 score threshold of 0.25, allowing for the grouping of similar responses.

In this context, the F1 score is calculated based on token inclusion : for two sequences $a = (a_1, \ldots, a_n)$ and $b = (b_1, \ldots, b_m)$, defining

$$p = \frac{|a \cap b|}{n} \quad \text{and} \quad r = \frac{|a \cap b|}{m}$$

Algorithm 2 Mutual Information Algorithm Used in [36]

Input:

$\mu \in \mathcal{M}_1(\mathcal{X})$	any distribution over $\mathcal{X}$
$k \in \mathbb{N}$	sample size
$\gamma_1, \gamma_2 \geq 0$	stabilization parameters (typically selected as $1/k$)
$s : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$	a similarity function
$\tau \in \mathbb{R}$	a similarity threshold

1. Independently sample outputs $X_1, \ldots, X_k \sim \mu \in \mathcal{M}_1(\mathcal{X})$

2. Construct a set of indices of unique elements $U = \{i \in [k] : X_i \neq X_j \quad \forall j < i\}$

3. Construct cluster centers $S \subset U$ according to the similarity function: for all $i, t \in S$, we have $s(X_i, X_t) < \tau$ and cluster associated with X_i is $D(i) = \{j \in U : s(X_i, X_j) \geq \tau\}$. Aggregated probabilities: for all $i, t \in S$,

$$\mu_1'(X_i) = \sum_{j \in D(i)} \mu(X_j), \quad \mu_2'(X_t|X_i) = \sum_{j \in D(t)} \mu(X_j|X_i)$$

4. Construct empirical distributions: for all $i, t \in S$,

$$\hat{\mu}_1(X_i) = \frac{\mu_1'(X_i)}{Z}, \quad \text{where} \quad Z = \sum_{j \in S} \mu_1'(X_j)$$

$$\hat{\mu}_2(X_t|X_i) = \frac{\mu_2'(X_t|X_i)}{Z_i}, \quad \text{where} \quad Z_i = \sum_{j \in S} \mu_2'(X_j|X_i)$$

$$\hat{\mu}(X_i, X_t) = \hat{\mu}_1(X_i)\hat{\mu}_2(X_t|X_i), \quad \hat{\mu}^{\otimes}(X_i, X_t) = \hat{\mu}_1(X_i) \sum_{j \in S} \hat{\mu}_1(X_j)\hat{\mu}_2(X_t|X_j)$$

5. Compute estimate

$$\hat{I}_k(\gamma_1, \gamma_2) = \sum_{i,t \in S} \hat{\mu}(X_i, X_t) \ln \left(\frac{\hat{\mu}(X_i, X_t) + \gamma_1}{\hat{\mu}^{\otimes}(X_i, X_t) + \gamma_2} \right)$$

(where $|a \cap b|$ is the size of the intersection of a and b, in which for repetitions of an element y, we consider the minimum number of repetitions in a and b, i.e.,

$$\min_{c \in \{a,b\}} |\{i : c_i = y\}|,$$

in calculating the size of the intersection) we define

$$\mathrm{F1} = \frac{2pr}{p+r}.$$

Relating to the standard definition of the F1 score, p and r play the role of precision and recall, respectively, if a is thought of as a prediction of b.

One crucial point is how the probabilities are computed. This aspect was not very clearly explained in the paper by [36], but I was able to replicate some of the results in the section on **The Best of Both Worlds**.

5.8 Using Mutual Information of answers with and without additional context to better focus

In the previous section, iterative prompting wich aims to measure the dependence of answers on the presence or absence of additional prompts (composed of adversarial or previous answers).

A related method, known as **Context-Aware Decoding** [25], has been developed to address the tendency of models to rely too heavily on prior knowledge while not focusing enough on the prompt's context.

Consider the query: `"Argentina won World Cups in 1978, 1986, and 2022. How many World Cups have Argentina won?"`

A model trained only until 2021 may predict the answer as `"Two"`, relying excessively on outdated knowledge. Context-aware decoding is designed to address this issue by appropriately balancing both prior knowledge and the immediate context.

The following distinctions can be made:

- **Prior knowledge**: Information learned during pretraining.
- **Contextual knowledge**: The specific information provided in the prompt, in this case, c : `"Argentina won World Cups in 1978, 1986, and 2022."`
- **Query knowledge**: The direct question being asked, in this case, x : `"How many World Cups have Argentina won?"`

The logits produced by querying the language model with and without the context are then combined. This process captures the subtle but significant shifts in the logit distribution caused by incorporating the context.

Just as before, the idea is to use the **mutual information** between the random variables representing the answers with and without the additional context. Excessive reliance on outdated knowledge—placing too much weight on prior information—can lead to hallucinations. To mitigate this, the logits are adjusted as follows:

$$y_t \sim \mathrm{softmax}\left[(1+\alpha) \cdot \mathrm{logit}_\theta(y_t \mid c, x, y_{<t}) - \alpha \cdot \mathrm{logit}_\theta(y_t \mid x, y_{<t})\right]$$

$$= \mathrm{softmax}\left[\mathrm{logit}_\theta(y_t \mid c, x, y_{<t}) - \alpha \times \mathrm{PMI}(y_t; c \mid x, y_{<t})\right]$$

The central picture of the paper can be found in 21 The key distinction from the method in [36] is that a **pointwise mutual information (PMI)** computation is performed, whereas [36] uses a full mutual information calculation. By using PMI, it becomes possible to use it as the final prediction, leveraging the principle that a reliable uncertainty measure can serve as the prediction itself.

This approach aslo contrasts with [36], where the issue arises from models relying too heavily on the context, leading to hallucinations when mutual information exceeds a certain threshold. In contrast, hallucinations in this method are attributed to over-reliance on outdated knowledge.

5.9 Shapley Value: Extensive Prompt Modification

Between the first subsection, where the additional prompt was adversarial, and the second, where the prompt was helpful and necessary, lies an approach that considers all possible combinations of the prompt. This idea is explored in [32], and can be understood through the lens of the Shapley Value. The objective of this subsection is to recall and introduce the formalism associated with this concept, before linking it to the previously discussed methods in this section.

Shapley Values. Consider a supervised learning model f trained on features $\mathbf{X} \in \mathcal{X} \subseteq \mathbb{R}^d$ to predict outcomes $Y \in \mathcal{Y} \subseteq \mathbb{R}$. The data are assumed to be distributed according to some fixed but unknown distribution $\mathcal{D}$. Shapley values are a feature attribution method that decomposes model predictions as a sum: $f(\mathbf{x}) = \phi_0 + \sum_{j=1}^{d} \phi(j, \mathbf{x})$, where ϕ_0 is the baseline expectation (i.e., $\phi_0 = \mathbb{E}_{\mathcal{D}}[f(\mathbf{x})]$), and $\phi(j, \mathbf{x})$ denotes the Shapley value of feature j at point $\mathbf{x}$.

To define this quantity, a value function $v : 2^{[d]} \times \mathbb{R}^d \to \mathbb{R}$ is required, which quantifies the payoff associated with subsets $S \subseteq [d]$ for a particular sample. This formalizes a cooperative game, where each feature is treated as a player. A common choice for defining payoffs in explainable AI (XAI) is as follows:

$$v_0(S, \mathbf{x}) := \mathbb{E}_{\mathcal{D}} \left[f(\mathbf{x}) \mid \mathbf{X}_S = \mathbf{x}_S \right],$$

where the complementary features $\overline{S}$ are marginalized with respect to the reference distribution $\mathcal{D}$. For any value function v, the following random variable represents the marginal contribution of feature j to coalition S at point $\mathbf{x}$:

$$\Delta_v(S, j, \mathbf{x}) := v(S \cup \{j\}, \mathbf{x}) - v(S, \mathbf{x}).$$

The Shapley value of feature j is then the weighted mean of this variable over all subsets:

$$\phi_v(j, \mathbf{x}) := \sum_{S \subseteq [d] \setminus \{j\}} \frac{|S|!(d - |S| - 1)!}{d!} \left[\Delta_v(S, j, \mathbf{x}) \right]. \tag{1}$$

However, from that [32] defined a certain type of value function v such that what the Shapley value measures will be the Mutual Information.

Theorem 8

Proposition from [32]:
Let us define

$$v_{IG}(S, \mathbf{x}) := -H(Y \mid \mathbf{x}_S),$$

and moreover:

$$v_{IG}(S, \mathbf{x}) = -v_H(S, \mathbf{x}) = -\mathbb{E}_{Y|\mathbf{x}_S} \left[v_L(S, \mathbf{x}) \right].$$

For all features $j \in [d]$, coalitions $S \subseteq [d] \setminus \{j\}$, and samples $\mathbf{x} \sim \mathcal{D}_{\mathcal{X}}$:

$$\Delta_{IG}(S, j, \mathbf{x}) = I(Y; x_j \mid \mathbf{x}_S).$$

Proof Substituting v_{IG} into the definition of $\Delta_v(S, j, \mathbf{x})$ gives:

$$\begin{aligned} \Delta_{IG}(S, j, \mathbf{x}) &= -H(Y \mid \mathbf{x}_S, x_j) + H(Y \mid \mathbf{x}_S) \\ &= H(Y \mid \mathbf{x}_S) - H(Y \mid \mathbf{x}_S, x_j) \\ &= I(Y; x_j \mid \mathbf{x}_S) \end{aligned}$$

$$= \int_{\mathcal{Y}} p(y, x_j \mid \mathbf{x}_S) \log \frac{p(y, x_j \mid \mathbf{x}_S)}{p(y \mid \mathbf{x}_S) p(x_j \mid \mathbf{x}_S)} \, dy.$$

In the penultimate line, the equality $I(Y;X) = H(Y) - H(Y \mid X)$ is exploited, which serves as the second definition of mutual information used in this report.

In the end, what matters, is that in either case, with Shapley values or Mutual Information, what is done is to subtract two distributions.

It can also be connected back to [36], if one consider the Shapley value as expressed by $\mathbb{E}_{Y|x_S}[\log \mathbb{P}(Y = y|x_S)]$, with x_S representing the set of prompt components. For instance, in the prompt `"What is the capital of the UK?"`, $\mathbb{P}$ refers to the probability distribution of the logits, and y is `"London"`. When considering $S \cup \{j\}$, the j can represent an adversarial prompt component, resulting in $I(Y; x_j \mid x_S)$ which is exactly what we measure in the case of $I(Y_1, Y_2)$ (in the adversarial prompting case).

Similarly, this concept can be tied back to **Context-Aware Decoding** by comparing the logit distributions with and without the context.

This type of method therefore dive deeper than solely the measure of the Mutual Information. By doing so, with more interpretable methods for uncertainty estimation, it can yield a 'second order' guarantee. Indeed, it will be possible to understand the reason of uncertainty of the model that might give interesting insights.

6 Best of both world

On one hand, the first section explored methods to measure uncertainty by manipulating the output of a single answer, particularly through semantic clustering. On the other hand, the second section focused on modifying the prompt to assess uncertainty by querying the LLM multiple times.

Finally, the last subsection of the previous section emphasized the importance of understanding the source of uncertainty.

6.0.1 Why is Mutual Information Better than Entropy?

And actually, what is happening is that Mutual Information is a better measure of Uncertainty since it can separate different sources of uncertainty. In [36], the authors found that their method based on **Mutual Information (M.I.)** outperformed the method developed by [18], known as **Semantic Entropy (S.E.)**, on certain types of datasets. The results of their comparison can be seen in 23. They evaluated the effectiveness of both methods in predicting when the model made correct decisions, focusing on the precision-recall curve. Here, recall refers to the percentage of queries where the method does not abstain, while precision measures the percentage of correct decisions among those queries.

Dealing with Multi-Label Queries Mutual Information performed better on datasets where multiple sources of uncertainty were present, specifically in cases involving multi-label queries, such as `Name a yellow fruit`.

For this type of question, there are numerous correct answers, such as `Banana` or `Lemon`. In these scenarios, the model should not converge to a single answer distribution, very peaked around just one word. Instead, as noted earlier, the distribution should remain unaffected by previous answers. Using Mutual Information between answers as a measure of uncertainty enhances the model's ability to handle uncertainty arising from the data itself. This type of noise is known as **data uncertainty** or sometimes **aleatoric uncertainty**.

Different Sources of Uncertainty As previously mentioned, uncertainty can arise from the data itself.

In response, [36], following the literature on uncertainty estimation [10], proposed categorizing queries into three distinct groups:

- **Queries with no uncertainty.** For example, the query `What is the capital of the UK?`. As we saw earlier, the model answers confidently, regardless of whether uncertainty is measured by semantic entropy or mutual information.

- **Queries with uncertainty arising from the data.** An example is the query `Name a yellow fruit.` Multiple answers are correct, such as `Lemon` or `Banana`. This type of uncertainty can only be measured through iterative prompting because the entropy may consistently remain above a certain threshold due to the variety of semantically distinct correct answers. This is known as **data uncertainty** or **aleatoric uncertainty**.
- **Queries with model uncertainty.** An example is the query `What is the national instrument of Ireland?`. Although there is only one correct answer, the model may fail to provide it. This is referred to as **model uncertainty** or **epistemic uncertainty**, which reflects a lack of knowledge on the part of the model.

6.1 Semantically clustering iterative answers

6.1.1 The Intuition

Mutual information and semantic entropy, although seemingly in competition, have been identified through extensive literature review as orthogonal concepts. As a result, they can be effectively combined.

These two approaches can be seen as orthogonal: one is prompt-based, while the other is purely logit-based. This distinction allows for a combination of both methods, as they operate at different levels of the model, providing a more comprehensive perspective on uncertainty.

The key idea is to retain the uncertainty measure proposed by [36], as it offers a more fine-grained assessment of uncertainty. The goal is to build on this by introducing a threshold condition for uncertainty. Specifically, the model's response should be:

$$I(Y_1, Y_2) > \text{threshold} \quad \Rightarrow \quad \text{LLM: “I do not know”}$$

While [36] focused on clustering answers to optimize model performance, the aim here is to incorporate semantic clustering into the process.

- **Step 1:** Introduce a threshold for mutual information such that $I(Y_1, Y_2) >$ threshold leads to an explicit declaration of uncertainty from the model, as suggested by [36].
- **Step 2:** Cluster the answers semantically for the Y_i's, following the approach described by [18].

This results in the following pseudo-code: Mutual Information Algorithm with Bidirectional Entailment Clustering 3.

6.1.2 Comparing With and Without Semantic Clustering

The initial algorithm from [36] was not shared, so we had to recreate it from scratch based on the pseudo-code. There were many implicit assumptions in the process. The first challenge was selecting the probability distribution μ. For this, we used the model's likelihood for a given sentence.

The second hidden assumption, which is not explicitly discussed in either [36] or [18], was determining the number of tokens to consider when comparing answers. Empirically, I chose to use 5 tokens. For example, while `Paris` may be equivalent to `It's Paris`, it would not be equivalent to `Paris A`. This issue arose in our experiments with **LLaMa 2** [12], as the model would sometimes not naturally end after the tokens that would be semantically similar.

Another important point is the use of temperature, as it influences the randomness of the model's output. Based on the discussion in [18], I decided to use a temperature of 0.7.

We selected four answers for the initial prompt, which corresponds to a first call to the generation model $\mathcal{M}_{\}]\backslash}$ with 4 outputs. For the NLI model $\mathcal{M}_{\mathcal{NLI}}$, a total of $4 \times 3 = 12$ inferences were made. For the second prompt, which needed to incorporate all the initial answers, this resulted in $4 \times 4 = 16$ additional iterations for $\mathcal{M}_{\}]\backslash}$. This process resulted in approximately 17 seconds of computation time per answer.

Due to these complexities, I did not have enough time to follow the traditional method of comparing approaches using AUROC or Precision-Recall metrics on a large question-answering dataset.

Algorithm 3 Our Mutual Information Algorithm with Bidirectional Entailment Clustering

Input:

$\mu \in \mathcal{M}_{gen}(\mathcal{X})$	any distribution over $\mathcal{X}$ which comes with a context x
$k \in \mathbb{N}$	sample size
$\gamma_1, \gamma_2 \geq 0$	stabilization parameters (typically selected as $1/k$)
$\tau \in \mathbb{R}$	a similarity threshold for clustering
NLI classifier $\mathcal{M}_{NLI}$	natural language inference model

1. Independently sample outputs $X_1, \ldots, X_k \sim \mu \in \mathcal{M}_gen(\mathcal{X})$
2. Construct a set of indices of unique elements $U = \{i \in [k] : X_i \neq X_j \quad \forall j < i\}$
3. Perform Bidirectional Entailment Clustering to form the clusters of answers:
for $2 \leq m \leq k$ **do**
  for $c \in \mathcal{C}$ **do**
    $X_c \leftarrow c_0$ ▷ Compare to already-processed meanings.
    $left \leftarrow \mathcal{M}_{NLI}(cat(x, X_c, "\langle g/\rangle", x, X_m))$ ▷ Use first sequence for each semantic class.
    $right \leftarrow \mathcal{M}_{NLI}(cat(x, X_m, "\langle g/\rangle", x, X_c))$ ▷ Does the old sequence entail the new one?
    if $left$ is entailment **and** $right$ is entailment **then** ▷ Vice versa?
      $c \leftarrow c \cup \{X_m\}$ ▷ Put into the existing class.
    end if
  end for
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{X_m\}$ ▷ Semantically distinct, gets own class.
end for
  Using the clustering result $\mathcal{C}$, construct the cluster centers $S \subset U$.
  For all $i, t \in S$, use $\mu(X_j)$ to define clusters based on entailment results.

4. Construct empirical distributions: for all $i, t \in S$,

$$\hat{\mu}_1(X_i) = \frac{\mu_1'(X_i)}{Z}, \quad \text{where} \quad Z = \sum_{j \in S} \mu_1'(X_j)$$

$$\hat{\mu}_2(X_t|X_i) = \frac{\mu_2'(X_t|X_i)}{Z_i}, \quad \text{where} \quad Z_i = \sum_{j \in S} \mu_2'(X_j|X_i)$$

$$\hat{\mu}(X_i, X_t) = \hat{\mu}_1(X_i)\hat{\mu}_2(X_t|X_i), \quad \hat{\mu}^{\otimes}(X_i, X_t) = \hat{\mu}_1(X_i) \sum_{j \in S} \hat{\mu}_1(X_j)\hat{\mu}_2(X_t|X_j)$$

5. Compute the estimate:

$$\hat{I}_k(\gamma_1, \gamma_2) = \sum_{i,t \in S} \hat{\mu}(X_i, X_t) \ln \left(\frac{\hat{\mu}(X_i, X_t) + \gamma_1}{\hat{\mu}^{\otimes}(X_i, X_t) + \gamma_2} \right)$$

Instead, we compared the methods by using questions from the three different groups of uncertainty: **No Uncertainty**, **Data Uncertainty**, and **Model Uncertainty**. To enhance the robustness of the results, each task was repeated 10 times per question, with each repetition referred to as a trial.

Here are the examples used for each group:

- **Group 1: No Uncertainty**
  - `What is the capital of the UK?`
  - `What is the largest country in the world?`
- **Group 2: Model Uncertainty**
  - `What is the national instrument of Ireland?`
  - `If Monday's child is fair of face, what is Saturday's child?`
- **Group 3: Data Uncertainty**
  - `Name a city in the UK.`
  - `Name a ball game that is played by more than 5 players.`

Mutual Information Algorithm with Bidirectional Entailment Clustering: Comparison of Different Types of Uncertainty The objective was to measure the mutual information for each of these questions across their different answers using our own algorithm (3). The results are presented in 5.

In the top panel, you can observe the mutual information values for the various questions. Each point represents a single trial, and the results are grouped into a box plot to better visualize the differences between the questions.

The bottom panel shows the p-values, which quantify the differences in mutual information between the different groups. The data was divided by quartile to remove the extreme values, though these extremes are still visible on the plot.

Replicating the Algorithm 2 As mentioned earlier, the algorithm from [36] was not shared, only a pseudo-code was provided. However, we were able to successfully replicate their method. The main difference from the previously presented algorithm was the use of the F_1 score.

The results are shown in 6. In the top panel, the mutual information values for the different questions are displayed. Each point represents a single trial, and the results are grouped into a box plot to better illustrate the differences between the questions.

On average, this method took 1.55 seconds per answer.

However, the mutual information values were extremely low, around $1e^{-7}$. This could lead to issues related to numerical precision, potentially causing negative mutual information values, which is theoretically impossible. This will be discussed further in the following paragraph.

Why Mutual Information should not be negative ? By definition,

$$I(X;Y) = -\sum_{x\in X}\sum_{y\in Y} p(x,y)\log\left(\frac{p(x)p(y)}{p(x,y)}\right)$$

Now, negative logarithm is convex and $\sum_{x\in X}\sum_{y\in Y} p(x,y) = 1$, therefore, by applying Jensen Inequality we will get,

$$I(X;Y) \geq -\log\left(\sum_{x\in X}\sum_{y\in Y} p(x,y)\frac{p(x)p(y)}{p(x,y)}\right)$$

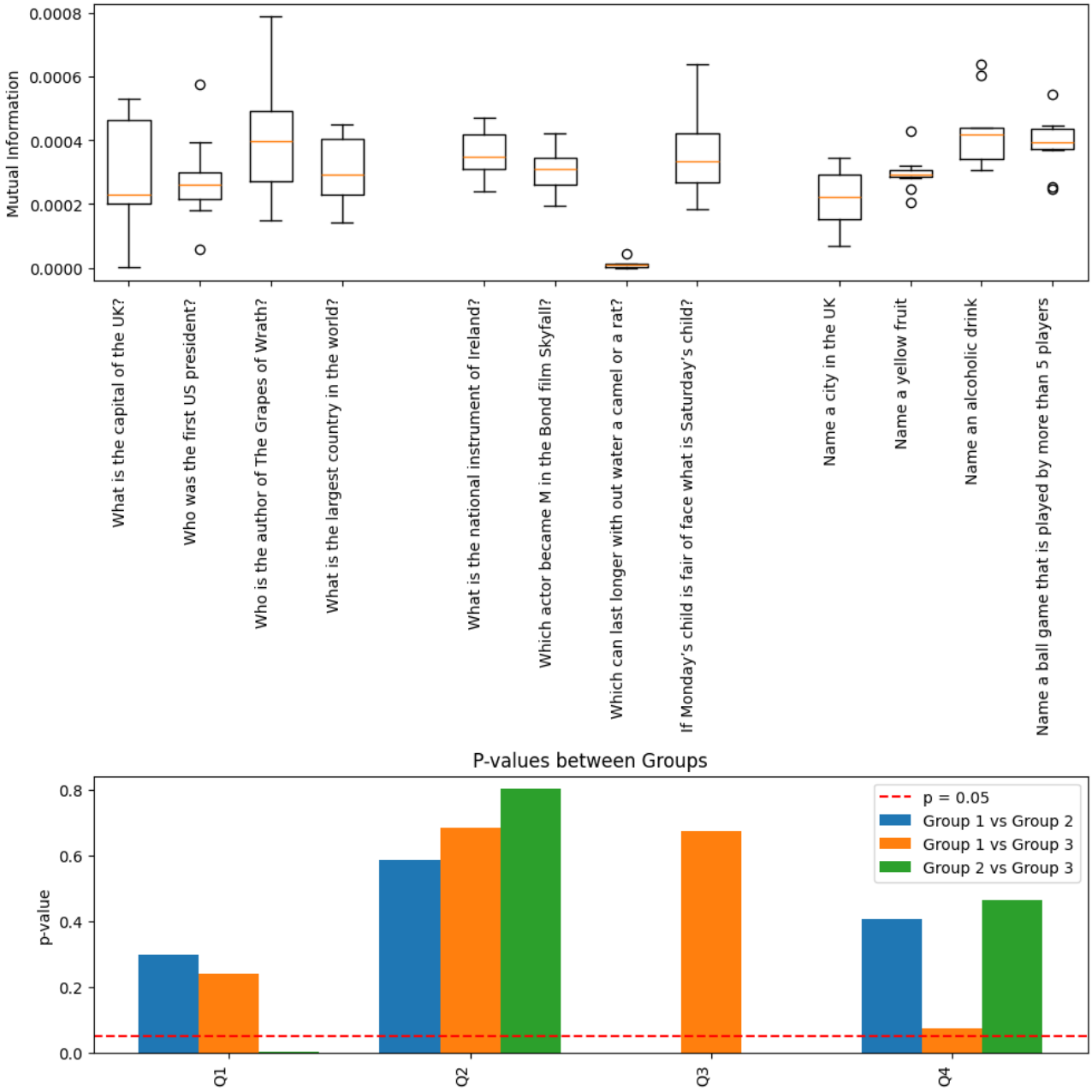


Figure 5: Mutual Information Algorithm with Bidirectional Entailment Clustering: Comparison of different types of uncertainty. 17 seconds on average per answer. The aim was to compare the mutual information of **Group 1: No Uncertainty**, **Group 2: Model Uncertainty**, **Group 3: Data Uncertainty**. It is possible to see significant differences between the groups when studied by quartile.

$$= -\log\left(\sum_{x\in X}\sum_{y\in Y} p(x)p(y)\right)$$

$$= 0$$

Q.E.D

However, since [36] did not share their final code or the exact Mutual Information values they used, it is possible that the numerical precision error we encountered may have also been present in their study. The way Mutual Information is used here is as a metric that only needs to exceed a certain threshold—it does not have to be an exact representation of true Mutual Information.

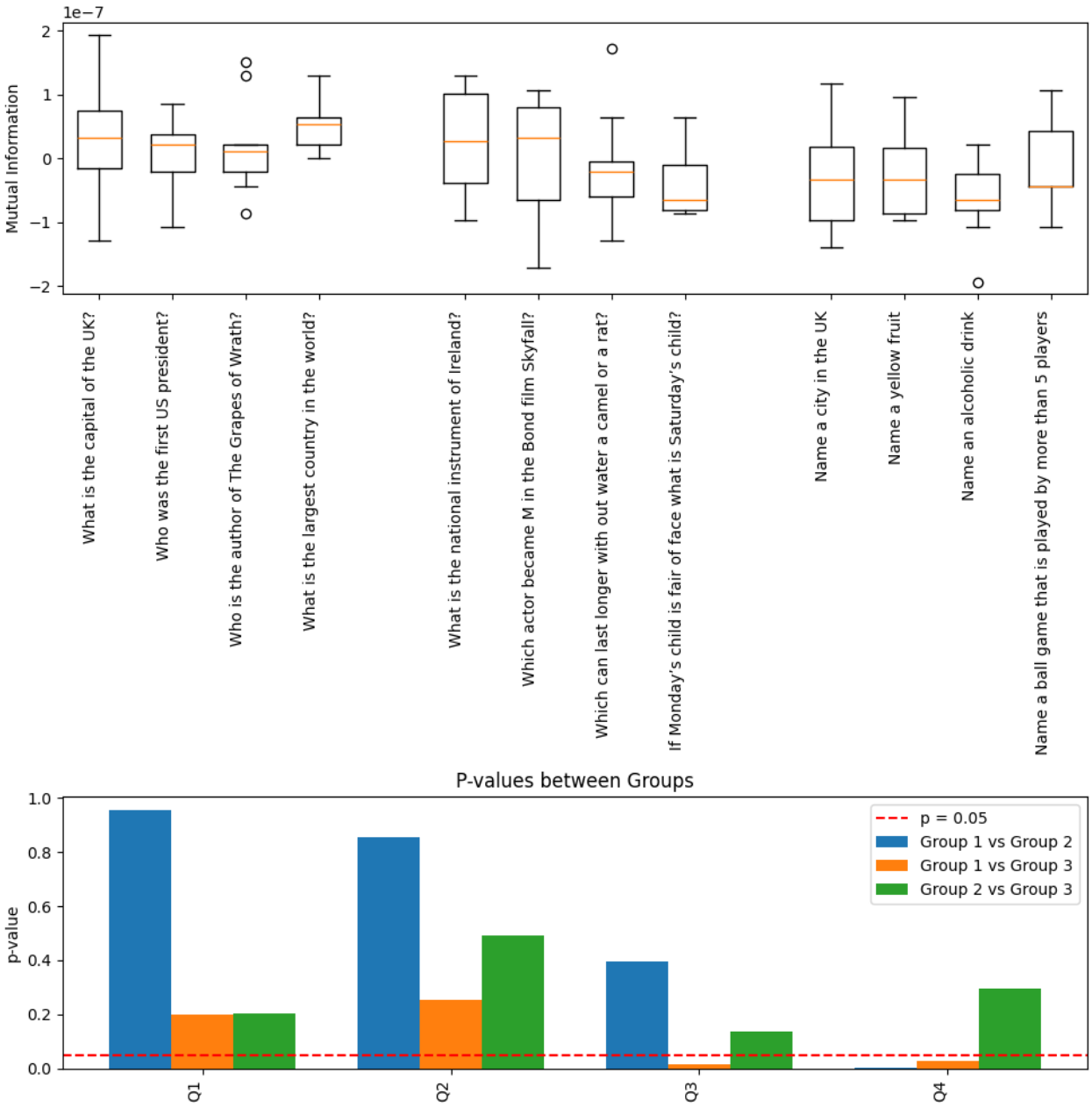


Figure 6: Mutual information with clustering by F1 score. 1.55 seconds per questions. The aim was to compare the mutual information of **Group 1: No Uncertainty**, **Group 2: Model Uncertainty**, **Group 3: Data Uncertainty**. It is possible to see significant differences between the groups when studied by quartile. Yet the values of the mutual information are extremely low $1e^{-7}$

Drawbacks and Advantages Ultimately, the method in 2 is approximately 10 times faster than the method we developed (3). At first glance, both methods perform similarly in separating the different groups of uncertainty. However, a key difference is the occurrence of negative Mutual Information values in the replicated method, which may be due to numerical precision issues. Thus, there is a trade-off between speed and accuracy. The choice between these two metrics will depend on the specific use case and the balance required between performance and precision.

6.2 Assessing the deviation of the semantic graphs during adversarial iterative prompting

On top of the combination of iterative prompting and the semantic entropy outlined above, another type of algorithm was explored.

Building on the intuition that led to the improvement of semantic entropy—namely, that the answers and their semantic relationships could be modeled as a graph—I sought to investigate the graph of responses within the context of adversarial iterative prompting. The underlying idea is that, as the adversarial prompt leads to an increase in incorrect tokens, the resulting graph of answers should increasingly diverge from the initial graph. By measuring the distance between the graphs, one could assess whether they progressively deviate from the original semantic graph as the adversarial prompting continues.

Summary of Iterative Prompting with Semantic Graph Evolution The algorithm was the following

- **1.** Do an iterative adversarial prompting technique just as before, for instance, adding `One possible answer is Paris.` to the query `What is the capital of the UK ?` iteratively
- **2.** Compute the relationships between answers using an NLI model ($\mathcal{M}_{NLI}(cat(x, a_i), cat(x, a_j)$, where x is the prompt and a_i,a_j are two answers).
- Store the answers with their respective likelihoods and the semantic relationship between them as a graph. If there is N iteration of the adversarial prompt, one should end up with a list of graphs `[G1, G2, ..., GN]`.
- Compute distances of the graphs `[G2, ..., GN]` to `G1`
- A measure of uncertainty that should be threshold can be either the maximum distance or the slope of the evolution of the distances to the first graph.

The remaining question is how to compute the distance between these graphs.
Two approaches were used from Optimal Transport.

6.2.1 Wasserstein Distance for Graphs

The first one is to use the Wasserstein Distance.

The Earth Mover's Distance (EMD), also known as the Wasserstein distance, is a measure of the distance between two probability distributions. The concept is inspired by the metaphor of moving "earth" (or mass) from one distribution to another. The EMD quantifies the minimum cost of transforming one distribution into another, where the cost is defined as the amount of earth moved multiplied by the distance it is moved.

Given two distributions, the EMD finds the optimal transport plan that minimizes the total cost of moving mass from one distribution to the other. This distance has wide applications, including image comparison, machine learning, and, as we explore here, graph comparison.

In the context of graphs, the Wasserstein distance can be used to compare two graphs by considering both their node features and edge structures. The objective is to find an optimal way to "move" the nodes and edges from one graph to the other with minimal cost.

Definition 9

Wasserstein Distance for Graphs:
Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs, where: - V_1 and V_2 are the sets of nodes, - E_1 and E_2 are the sets of edges.
Each node $v_i \in V_1$ and $v_j \in V_2$ is associated with a feature vector x_{v_i} and x_{v_j}, respectively (the likelihood of the answers in our case). Each edge $e_{i,j} \in E_1$ and $e_{i,j} \in E_2$ is associated with a weight or feature $w_{i,j}$ and $w'_{i,j}$, respectively.
The Wasserstein distance $d_W(G_1, G_2)$ between two graphs is given by the optimal transport formulation:

$$d_W(G_1, G_2) = \min_{\pi \in \Pi(V_1, V_2)} \left(\sum_{i \in V_1} \sum_{j \in V_2} \pi_{i,j} \, C_V(v_i, v_j) + \sum_{(i,k) \in E_1} \sum_{(j,l) \in E_2} \pi_{i,j} \pi_{k,l} \, C_E(e_{i,k}, e_{j,l}) \right)$$

where:

- $\Pi(V_1, V_2)$ is the set of all possible transport plans, i.e., joint probability distributions π over $V_1 \times V_2$ that satisfy:
$$\sum_{j \in V_2} \pi_{i,j} = p_{v_i}, \quad \sum_{i \in V_1} \pi_{i,j} = p_{v_j},$$
where p_{v_i} and p_{v_j} are the node probabilities (masses) for $v_i \in V_1$ and $v_j \in V_2$.
- $C_V(v_i, v_j)$ is the cost of transporting node $v_i \in V_1$ to node $v_j \in V_2$, typically defined as:
$$C_V(v_i, v_j) = \|x_{v_i} - x_{v_j}\|_p^p,$$
where $\| \cdot \|_p$ is the p-norm (often $p = 1$ or $p = 2$) over the node feature vectors.
- $C_E(e_{i,k}, e_{j,l})$ is the cost of transporting edge $e_{i,k} \in E_1$ to edge $e_{j,l} \in E_2$, which could be defined as the difference in edge weights or some other measure of similarity:
$$C_E(e_{i,k}, e_{j,l}) = \|w_{i,k} - w'_{j,l}\|_p^p.$$

The goal is to find the optimal transport plan π that minimizes the total cost of transforming the node and edge distributions of G_1 into G_2.

The idea was then to try to compare the different groups of Uncertainty with this methods. The following could be depicted 7. The Y-axis is the distance, and the x-axis is the index, which correspond to the number of repetition of the adversary prompt. Each line correspond to one query. The color represent the type of uncertainty. Unfortunately, whether the question were with No Uncertainty, Model Uncertainty or Data Uncertainty no differences seems to have appear.

6.2.2 Gromov-Wasserstein Distance for Graphs

This motivated us to try a new distance which is the Gromov-Wasserstein distance. The Gromov-Wasserstein distance generalizes the Wasserstein distance to compare two metric spaces (or graphs) based on their structure.

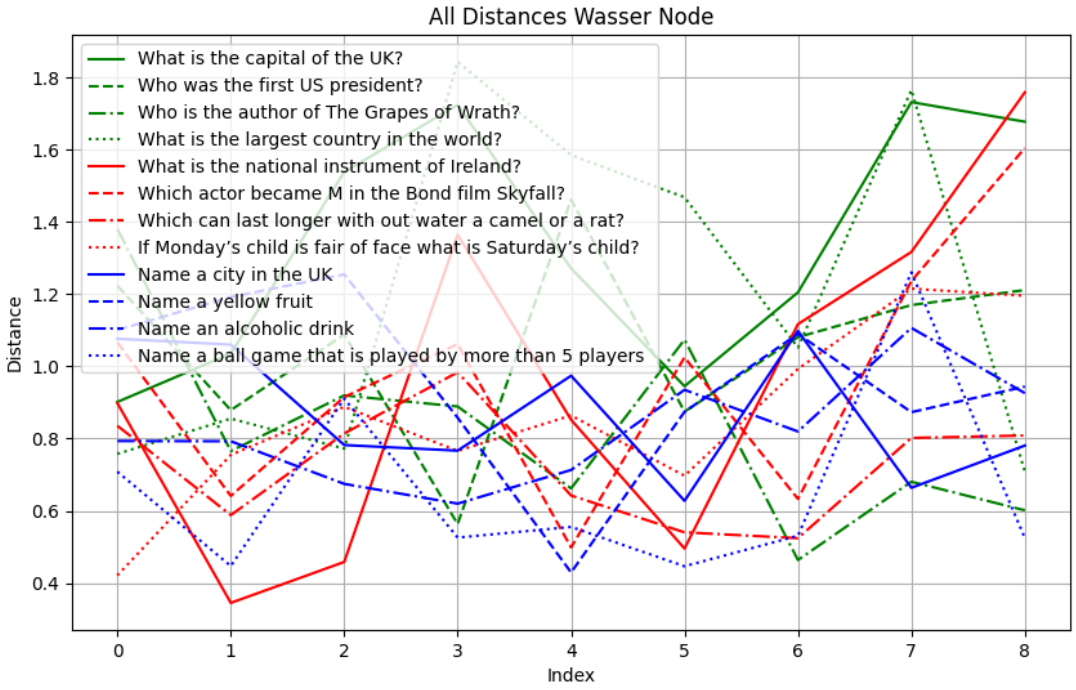


Figure 7: **Evolution of the Wasserstein distance of the Semantic graphs during iterative prompting.** The Y-axis is the distance, and the x-axis is the index, which correspond to the number of repetition of the adversary prompt. Each line correspond to one query. The color represent the type of uncertainty. Unfortunately, whether the question were with No Uncertainty, Model Uncertainty or Data Uncertainty no differences seems to have appear

Definition 10

Gromov Wasserstein distance:
Given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, the Gromov-Wasserstein distance between them is defined as:

$$GW(G_1, G_2) = \inf_{\gamma \in \Gamma(\mu_1, \mu_2)} \sum_{(v_1, v_2) \in V_1 \times V_2} \sum_{(v_1', v_2') \in V_1 \times V_2} |d_1(v_1, v_1') - d_2(v_2, v_2')|^2 \, d\gamma(v_1, v_2) \, d\gamma(v_1', v_2')$$

where:

- $d_1(v_1, v_1')$ is the distance between nodes v_1 and v_1' in graph G_1,
- $d_2(v_2, v_2')$ is the distance between nodes v_2 and v_2' in graph G_2,
- γ is a coupling between the two graphs,
- $\Gamma(\mu_1, \mu_2)$ is the set of all possible couplings between the node distributions μ_1 and μ_2.

And just as before, unfortunately, no difference can be seen between the three groups.

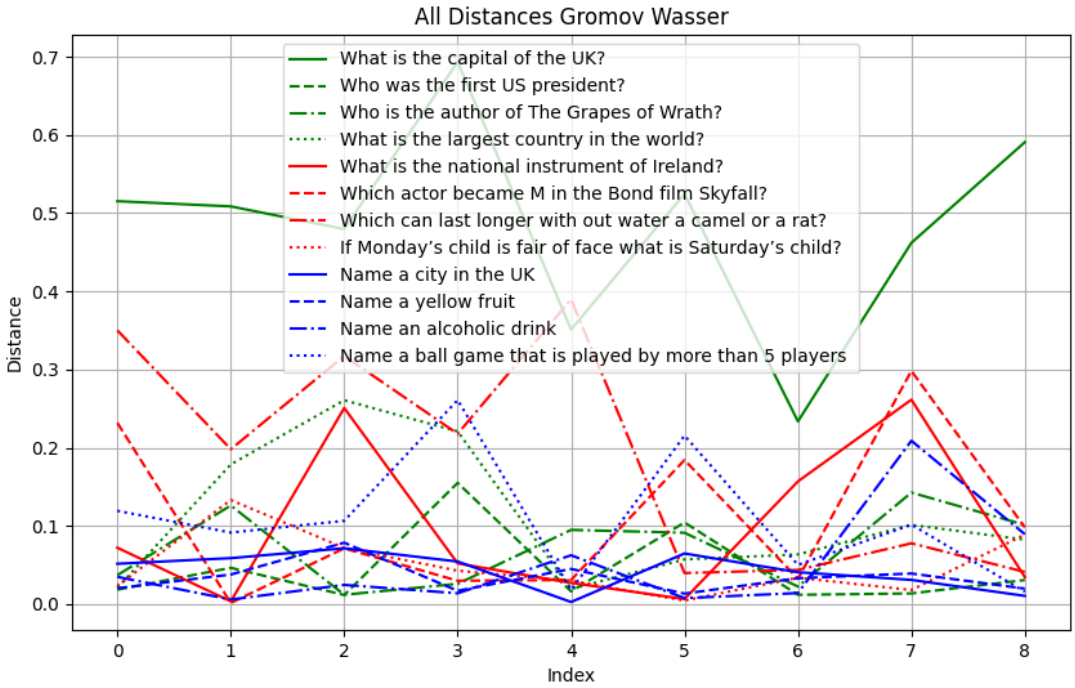


Figure 8: **Evolution of the Gromov-Wasserstein distance of the Semantic graphs during iterative prompting.** The Y-axis is the distance, and the x-axis is the index, which correspond to the number of repetition of the adversary prompt. Each line correspond to one query. The color represent the type of uncertainty. Unfortunately, whether the question were with No Uncertainty, Model Uncertainty or Data Uncertainty no differences seems to have appear

7 Discussion

7.1 Future Work and Perspectives

Better Assessing the Newly Developed Algorithm As mentioned in the previous section, there was not enough time to fully evaluate the developed methods. The immediate next step will be to assess 3 using either AUROC or precision-recall curves on datasets such as TriviaQA [14] and WordNet [8], which contains multi-label queries. Once this evaluation is complete, it could be beneficial to fine-tune the model's hyperparameters, particularly the maximum number of tokens per answer, which impacts the semantic clustering. Additionally, it would be valuable to explore fine-tuning the model's temperature settings.

Assessing the performance of the method on larger Q&A datasets could also extend to evaluating the deviation of the semantic graph during adversarial iterative prompting. For this method, it would be interesting to experiment with different distance metrics to measure changes in the graph. One possible approach could be applying topological data analysis by transforming the graphs into Maximum Spanning Trees and then creating birth-death diagrams. These diagrams could be treated as distributions, which could then be compared using Wasserstein distances [3].

Combining Mutual Information with Entropy Another proposed approach is to compute both Semantic Entropy and Mutual Information for a given query through iterative prompting. The goal would be to combine these two measures and then apply a bi-dimensional classifier to determine whether the model should respond with confidence or indicate uncertainty. This combined approach could provide a more robust assessment of the model's uncertainty.

7.2 Combining Not Only the Prompt and Logits but Reaching for the Most Distant Layer

In this report, all methods (except for the brief remark on Entropy Neurons) were designed for models where only the input and the penultimate layer (logits) are accessible. However, in a scenario where one has access to all layers of the model, additional alternatives could lead to interesting improvements.

Decoding by Contrasting Layers The DoLA (Decoding by Contrasting Layers) algorithm begins by identifying the layer that is furthest from the final layer in terms of the Jensen-Shannon Divergence (JSD), a symmetric measure of distribution distance. Once the layer with the least similarity to the final layers is determined, its influence is subtracted from the final result. When two tokens are in conflict (i.e., they have similar probabilities), the algorithm assesses which token is more sensitive to changes within the network.

This process is somewhat analogous to measuring the mutual entropy between the last layers. The equation for DoLA is as follows:

We define

$$M = \arg\max_{j \in J} \text{JSD}(q_N(|x_{<t})||q_j(|x_{<t})),$$

where $q_j(x_t|x_{<t}) = \text{softmax}(\phi(h_t^{(j)}))_{x_t}$ and $h_t^{(j)}$ is the embedding at layer j.

Then,

$$\hat{p}(x_t|x_{<t}) = \text{softmax}(F(q_N(x_t), q_M(x_t)))_{x_t},$$

where

$$F(q_N(x_t), q_M(x_t)) = \begin{cases} \log\left(\frac{q_N(x_t)}{q_M(x_t)}\right), & \text{if } x_t \in \mathcal{V}_{head}(x_t|x_{<t}) \\ -\infty, & \text{otherwise.} \end{cases}$$

Here, $\mathcal{V}_{head}(x_t|x_{<t}) = \{x_t \in \mathcal{X} : q_N(x_t) \geq \alpha \max_w(q_N(w))\}$.

This method closely resembles Mutual Information (MI). While JSD is typically used to compare the similarity between distributions, MI is employed to quantify and understand the relationship between variables.

24

7.3 A Mathematical Limit on Mutual Information

The perspectives proposed so far have been more focused on the engineering side, with less attention to the mathematical aspects. However, there is an important point to address regarding one of the core measures in this report—Mutual Information (MI)—as a measure of uncertainty.

The concept of using Mutual Information to address epistemic uncertainty was first introduced in [1]. The total uncertainty of a random variable Y, represented by its entropy $H(Y)$, can be decomposed as:

$$H(Y) = H(Y|\Theta) + I(Y;\Theta),$$

where $H(Y|\Theta)$ represents aleatoric uncertainty, capturing the uncertainty in Y given fixed model parameters Θ, and $I(Y;\Theta)$ quantifies epistemic uncertainty, measuring how much information the model parameters Θ provide about the outcome Y. Thus, aleatoric uncertainty reflects inherent randomness in the system, while epistemic uncertainty arises from uncertainty in the model's knowledge of the parameters.

Since then, Mutual Information has been widely used as a measure of epistemic uncertainty. In this report, we applied Mutual Information between answers, not between the model and its parameters. However, I believe that the underlying concept remains quite similar.

Recently, [33] raised an important point, questioning the appropriateness of MI as a measure of uncertainty.

Mutual Information as a Measure of Conflict As shown in 25, which depicts the distribution over a model's learned parameters, there is some incoherence in using Mutual Information (specifically

between outcomes and parameters) as a measure of uncertainty.

In panel (a), a flat distribution over the model parameters is presented, and in panel (f), two "semi-Dirac" distributions are shown. Interestingly, the Epistemic Uncertainty (EU) of the flat distribution is 0.28, which is lower than the EU value for the two parameters in panel (f), which is 1. Intuitively, one might think (and I agree) that a model is more uncertain if it places a uniform prior over its parameter distribution rather than having just two distinct opposing parameters.

The authors of [33] argue that Mutual Information may be better interpreted as a measure of conflict between representations (or model parameters) rather than as an indicator of knowledge deficiency.

Conclusion: MI as a Measure of Conflict In the end, it seems that what matters is measuring conflict, or Mutual Information, between distributions with and without context—at least in cases such as Csaba's method, DoLA, and CAD.

7.4 Why Are Lack of Knowledge and Conflict So Often Related?

The intertwining of lack of knowledge and conflict led me to consider the possible reasons for this connection.

One potential explanation comes from the linguistic interpretation of the word "certainty," which has two distinct etymological roots. The first is the Latin word *cerno*, meaning "to separate," as in the phrase "separate the wheat from the chaff." This relates to uncertainty as a conflict that needs to be separated and resolved. The other root is *circinare*, which referred to the act of drawing a little circular trench around an olive tree to keep water near its roots, symbolizing the drawing of an inside and the outside. This relates more to uncertainty as a lack of knowledge, not knowing where to draw this circle between the inside and the outside. Language is a model of the world, and thus we use words (or converge on words) that are useful. Thus, there might be a link why these two forms of uncertainty might have merged toegether.

Until now, words have been exclusively used by Homo sapiens, so it's possible that this convergence of meanings evolved from a shared cognitive basis. In cognitive science, the decision-making process in animals, including humans, is often modeled using the drift-diffusion model (DDM), depicted in 26 [23]. In this model, decisions are made when enough information is accumulated from the environment to choose one option over another. Thus, a lack of knowledge is useful only insofar as it helps resolve a conflict between two competing choices.

Final Thoughts I find it fascinating that in the pursuit of developing mathematical models for language that can express uncertainty—such as saying `I do not know`—we encounter mathematical inconsistencies and counterexamples that point to a deeper question: why have humans, through time, conflated a verb used for the separation of the wheat (*cerno*) with one used to describe a technic to have more hydrated olive trees (*circinare*).

Bon app'

8 Bibliography

References

[1] Robert B. Ash. Information Theory. Dover Publications, New York, 1965.

[2] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, [...] Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[3] Frédéric Chazal and Bertrand Michel. An introduction to topological data analysis: fundamental and practical aspects for data scientists, 2021.

[4] Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James Glass, and Pengcheng He. Dola: Decoding by contrasting layers improves factuality in large language models, 2024.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[6] Ekaterina Fadeeva, Roman Vashurin, Akim Tsvigun, Artem Vazhentsev, Sergey Petrakov, Kirill Fedyanin, Daniil Vasilev, Elizaveta Goncharova, Alexander Panchenko, Maxim Panov, Timothy Baldwin, and Artem Shelmanov. Lm-polygraph: Uncertainty estimation for language models, 2023.

[7] Sebastian Farquhar, Jannis Kossen, Lukas Kuhn, and Yarin Gal. Detecting hallucinations in large language models using semantic entropy. Nature, 630(8017):625–630, 2024.

[8] Christiane Fellbaum. WordNet: An Electronic Lexical Database. MIT Press, 1998.

[9] Stephen M. Fleming. Know Thyself: The Science of Self-awareness. Basic Books, illustrated edition, 2021.

[10] Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, Muhammad Shahzad, Wen Yang, Richard Bamler, and Xiao Xiang Zhu. A survey of uncertainty in deep neural networks, 2022.

[11] Neha Gupta, Harikrishna Narasimhan, Wittawat Jitkrittum, Ankit Singh Rawat, Aditya Krishna Menon, and Sanjiv Kumar. Language model cascades: Token-level uncertainty and beyond, 2024.

[12] [..]Yuchen Zhang Hugo Touvron, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

[13] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023.

[14] Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension, 2017.

[15] Saurav Kadavath, [..] Chris Olah, and Jared Kaplan. Language models (mostly) know what they know, 2022.

[16] R. I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete structures. In Proceedings of the 19th International Conference on Machine Learning, volume 2002, pages 315–322, 2002.

[17] Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Clam: Selective clarification for ambiguous questions with generative language models, 2023.

[18] Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation, 2023.

[19] Zhen Lin, Shubhendu Trivedi, and Jimeng Sun. Generating with confidence: Uncertainty quantification for black-box large language models, 2024.

[20] Andrey Malinin and Mark Gales. Uncertainty estimation in autoregressive structured prediction, 2021.

[21] Ramishah Maruf. Lawyer apologizes for fake court citations from chatgpt. CNN Business, May 2023. Accessed: 2024-09-15.

[22] Alexander Nikitin, Jannik Kossen, Yarin Gal, and Pekka Marttinen. Kernel language entropy: Fine-grained uncertainty quantification for llms from semantic similarities, 2024.

[23] R. Ratcliff and G. McKoon. The diffusion decision model: theory and data for two-choice decision tasks. Neural Computation, 20(4):873–922, Apr 2008.

[24] Timo Schick and Hinrich Schütze. Exploiting cloze questions for few shot text classification and natural language inference, 2021.

[25] Weijia Shi, Xiaochuang Han, Mike Lewis, Yulia Tsvetkov, Luke Zettlemoyer, and Scott Wen tau Yih. Trusting your evidence: Hallucinate less with context-aware decoding, 2023.

[26] Alessandro Stolfo, Ben Wu, Wes Gurnee, Yonatan Belinkov, Xingyi Song, Mrinmaya Sachan, and Neel Nanda. Confidence regulation neurons in language models, 2024.

[27] R. J. Strawbridge, J. Ward, L. M. Lyall, et al. Genetics of self-reported risk-taking behaviour, trans-ethnic consistency and relevance to brain gene expression. Translational Psychiatry, 8:178, 2018.

[28] Joshua Strong, Qianhui Men, and Alison Noble. Towards human-ai collaboration in healthcare: Guided deferral systems with large language models, 2024.

[29] Gemini Team. Gemini: A family of highly capable multimodal models, 2024.

[30] Vicuna. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, 2023. Accessed: 2024-09-15.

[31] Ulrike von Luxburg. A tutorial on spectral clustering, 2007.

[32] David S. Watson, Joshua O'Hara, Niek Tax, Richard Mudd, and Ido Guy. Explaining predictive uncertainty with information theoretic shapley values, 2023.

[33] Lisa Wimmer, Yusuf Sale, Paul Hofman, Bern Bischl, and Eyke Hüllermeier. Quantifying aleatoric and epistemic uncertainty in machine learning: Are conditional entropy and mutual information appropriate measures?, 2023.

[34] Danny Wood, Theodore Papamarkou, Matt Benatan, and Richard Allmendinger. Model-agnostic variable importance for predictive uncertainty: an entropy-based approach, 2024.

[35] BigScience Workshop. Bloom: A 176b-parameter open-access multilingual language model, 2023.

[36] Yasin Abbasi Yadkori, Ilja Kuzborskij, András György, and Csaba Szepesvári. To believe or not to believe your llm, 2024.

9 Appendix

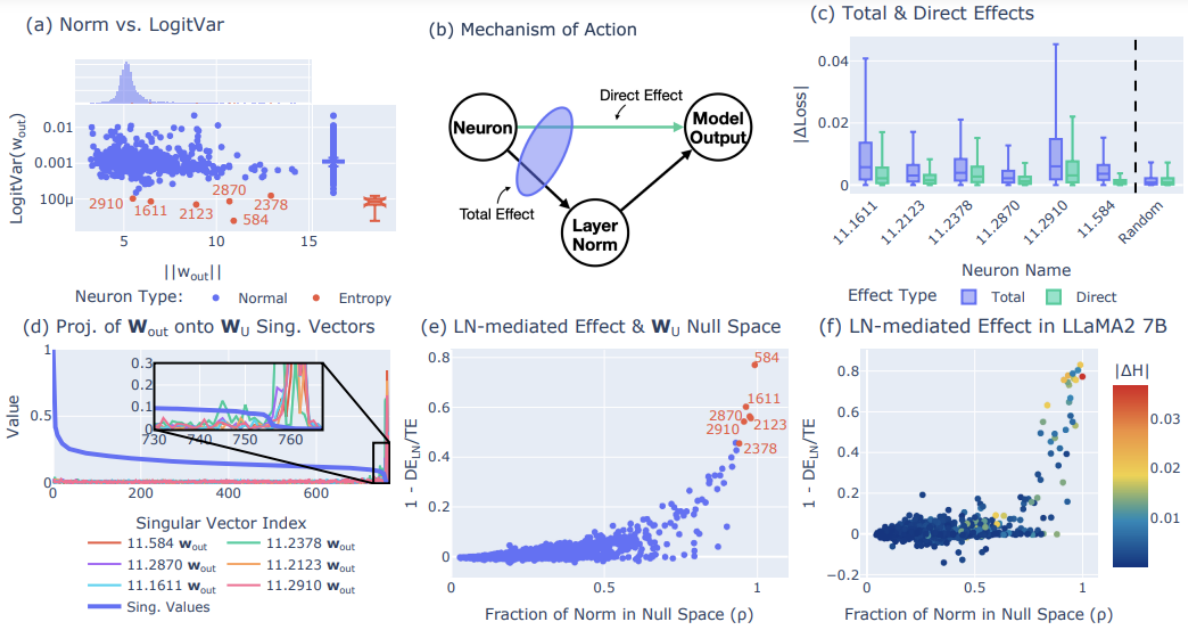


Figure 2: **Identifying and Analyzing Entropy Neurons.** (a) Neurons in GPT-2 Small displayed by their weight norm and variance in logit attribution. Entropy neurons (red) have high norm and low logit variance. (b) Causal graph showing the total effect and direct effect (bypassing LayerNorm) of a neuron on the model's output. (c) Comparison of total and direct effects on model loss for entropy neurons and randomly selected neurons. (d) Singular values and cosine similarity between neuron output weights and singular vectors of $\mathbf{W}_{\mathrm{U}}$. (e) Entropy neurons (red) show significant LayerNorm-mediated effects and high projection onto the null space (ρ). (f) Relationship between ρ and the LayerNorm-mediated effect in LLaMA2 7B. ρ is computed with $k = 40 \approx 0.01 * d_{\text{model}}$. Color represents absolute change in entropy upon ablation (ΔH).

Figure 9: Illustration of entropy neurons and their impact on the token distribution.

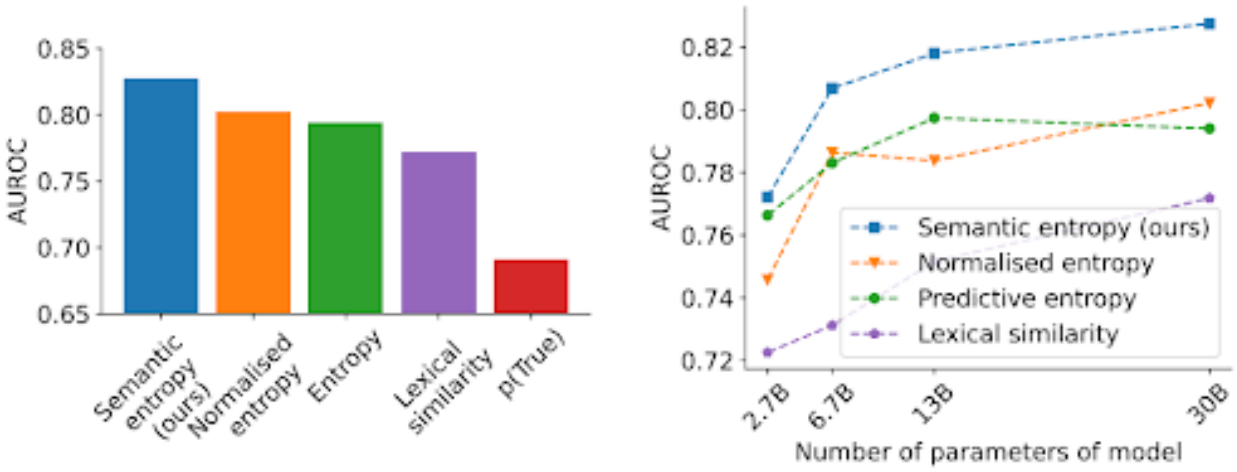


Figure 10: Using the TriviaQA dataset, the Semantic entropy yields a higher AUROC than the entropy not clustered, and this indepently of the model size. Figure from [18]

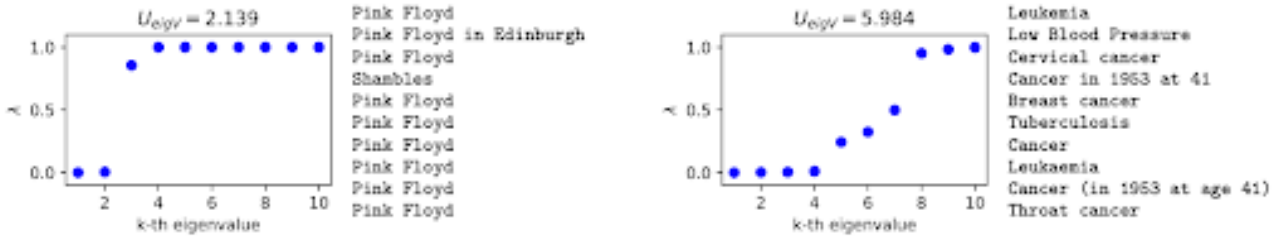


Figure 11: For the question on Pink Floyd, the model only give two groups of semantic answers. On the other, when asked about cancer there are much more different answers. As it was shown in Theorem 3, there is a link between the 0 eigen vectors of the Laplacian and the number of connected components of a graph. This is why there are more eigen vectors in the null space in the right panel. Figure from [19]

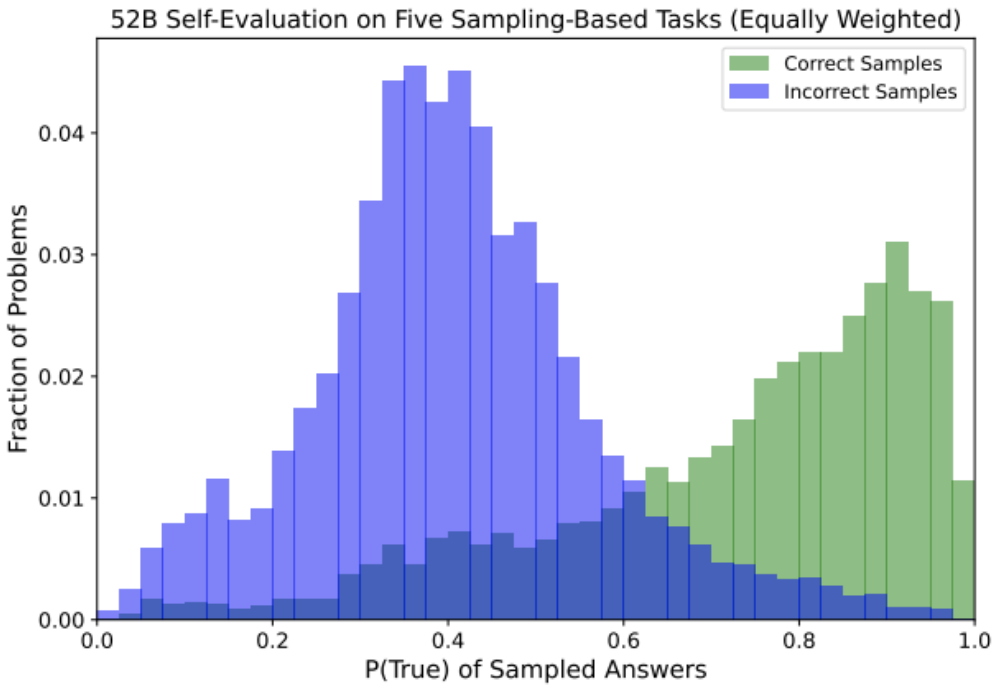
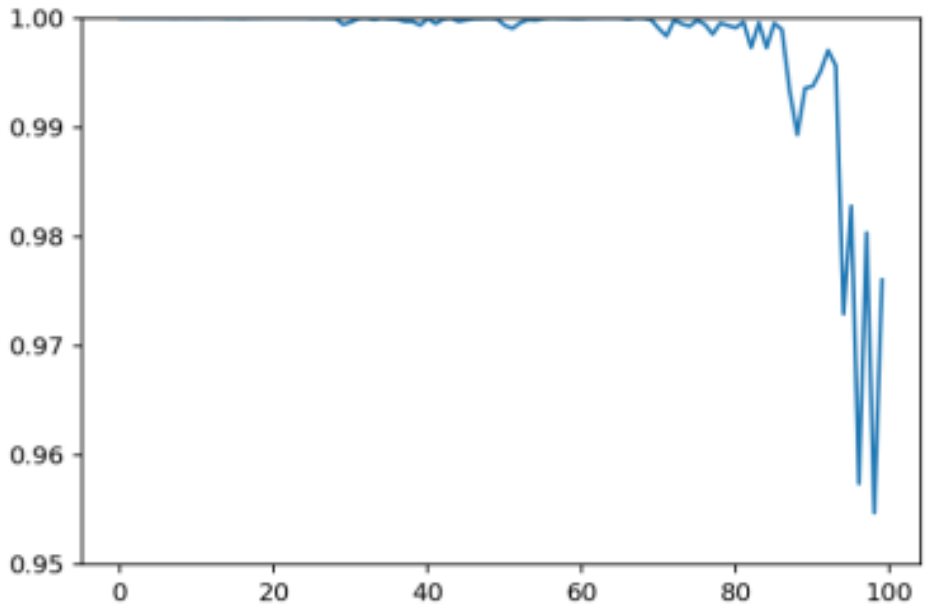


Figure 12: When formatted into a Yes-No Question, the model Claude of Anthropic is able to distinguish correct from wrong answers by giving more or less probability to the token `True`. Figure from [15]



Q: *What is the capital of the UK?*
A: *London* ($\approx$ 1.0) and *Paris* (1.29×10^{-10}).

Figure 13: Figure from [18]. The X-axis is the repetition of the wrong token (`Paris`) and the Y-axis is the probability of the correct answer (`London`). The probability of the good token `London` is not impacted by the repetition of the wrong answer `Paris`. The model is confident.

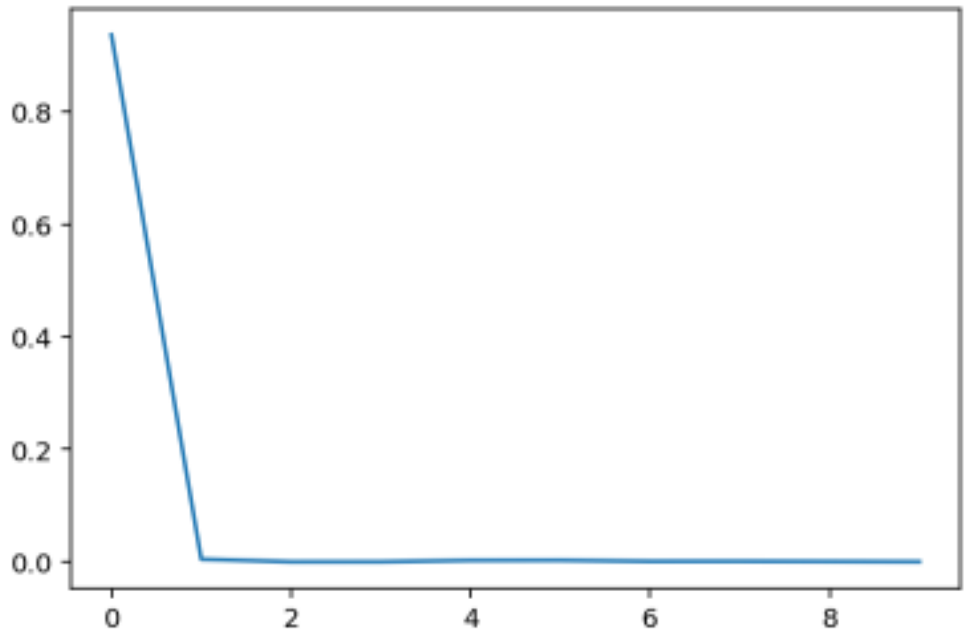


Figure 14: Figure from [18]. The X-axis is the repetition of the wrong answer (`Uilleann pipes`) and the Y-axis is the probability of the correct answer (`The harp`). The probability of the good token `The harp` is higly impacted by the repetition of the wrong answer `Uilleann pipes`. The model is not confident

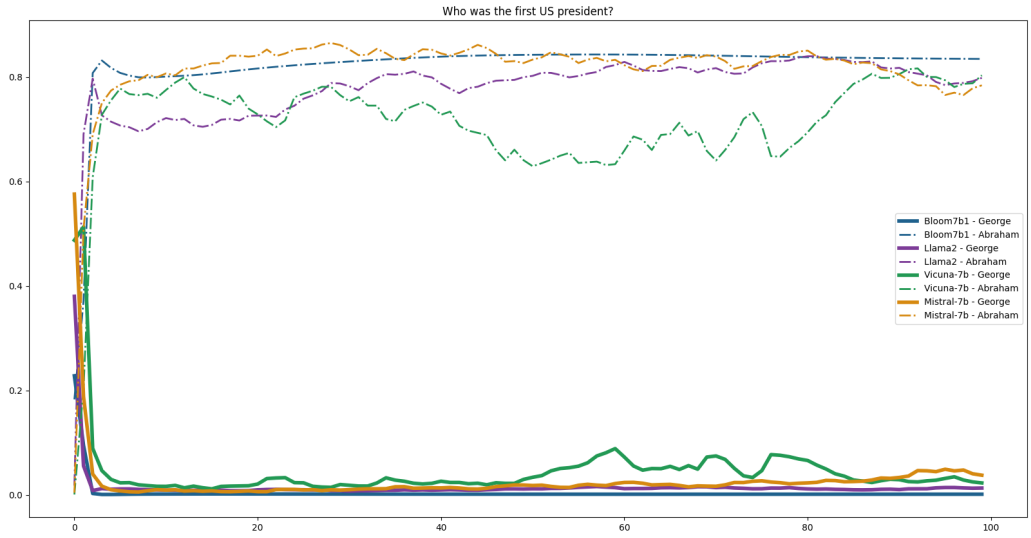


Figure 15: The X-axis represents the repetition of the adversary answer, `Abraham Lincoln`, to the question `Who was the first president of the United States`. The Y-axis represent the probability of tokens, either the good answer, `George Washingthon`, in bold line, or the wrong one, `Abraham Lincoln`, in dashed line. Each color pair corresponds to a different LLM. The prompt used was the same as the one initially proposed by [36].

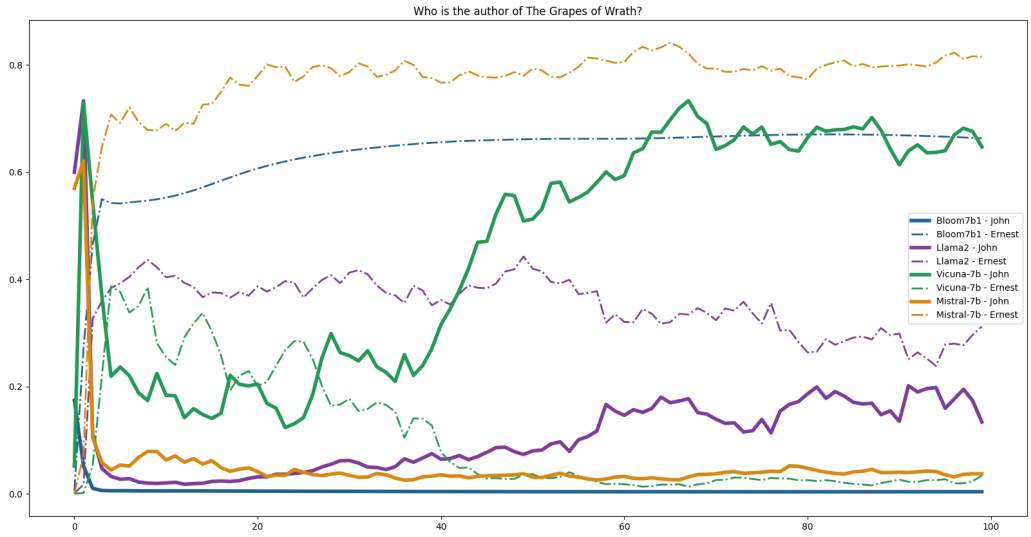


Figure 16: The X-axis represents the repetition of the adversary answer, `Ernest Hemingway`, to the question `Who is the author of The Grapes of Wrath?`. The Y-axis represent the probability of tokens, either the good answer, `John Steinbeck`, in bold line, or the wrong one, `Ernest Hemingway`, in dashed line. Each color pair corresponds to a different LLM. The prompt used was the same as the one initially proposed by [36].

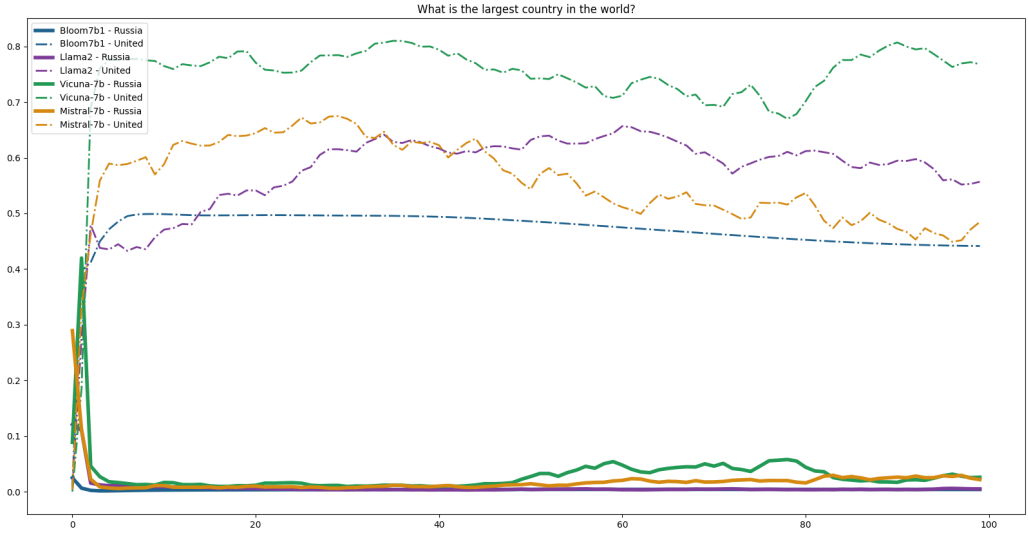


Figure 17: The X-axis represents the repetition of the adversary answer, `UK`, to the question `What is the largest country in the world?`. The Y-axis represent the probability of tokens, either the good answer, `Russia`, in bold line, or the wrong one, `UK`, in dashed line. Each color pair corresponds to a different LLM. The prompt used was the same as the one initially proposed by [36].

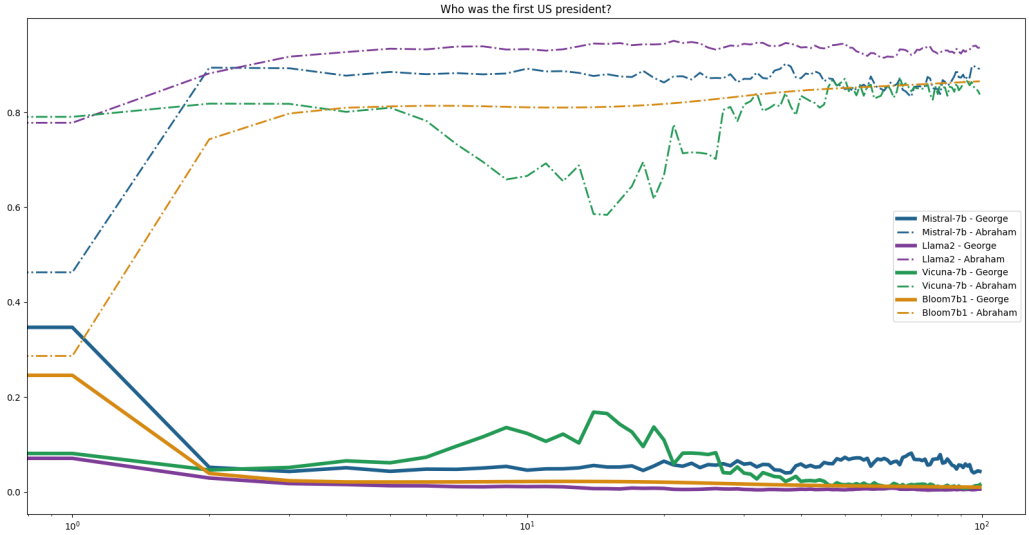


Figure 18: The X-axis represents the repetition of the adversary answer, `Abraham Lincoln`, to the question `Who was the first president of the United States`. The Y-axis represent the probability of tokens, either the good answer, `George Washingthon`, in bold line, or the wrong one, `Abraham Lincoln`, in dashed line. Each color pair corresponds to a different LLM. The prompt used was different from the one initially proposed by [36].

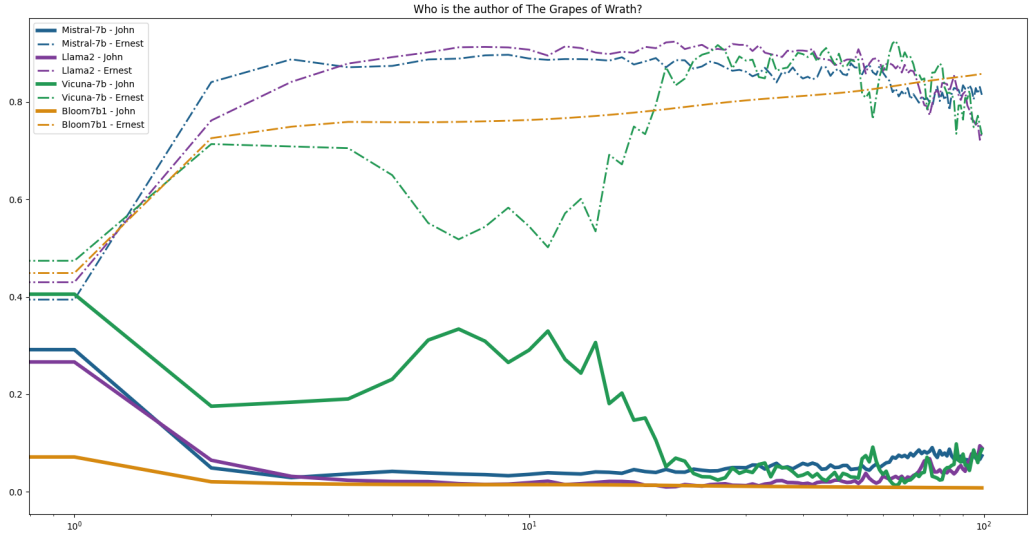


Figure 19: he X-axis represents the repetition of the adversary answer, `Ernest Hemingway`, to the question `Who is the author of The Grapes of Wrath?`. The Y-axis represent the probability of tokens, either the good answer, `John Steinbeck`, in bold line, or the wrong one, `Ernest Hemingway`, in dashed line. Each color pair corresponds to a different LLM. The prompt used was different from the one initially proposed by [36].

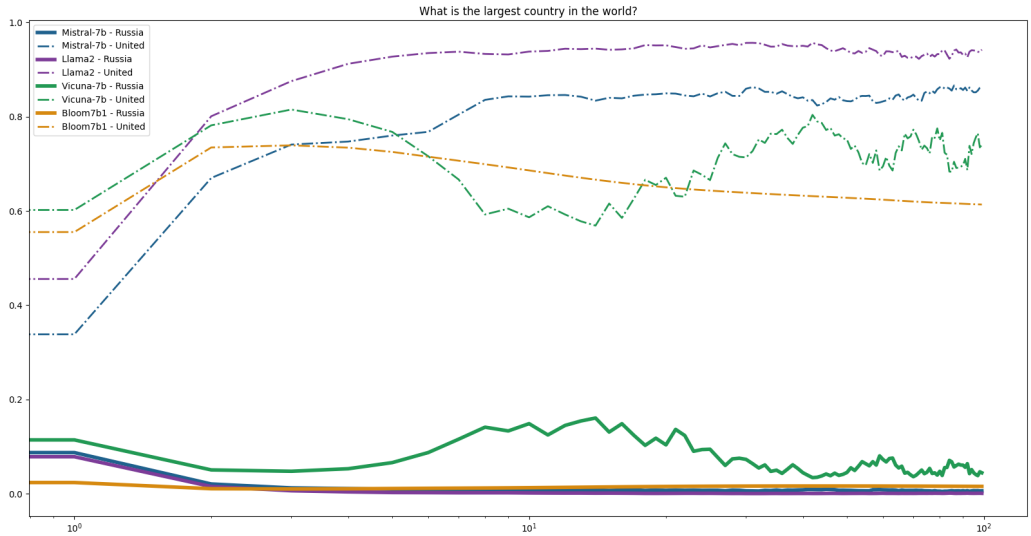


Figure 20: The X-axis represents the repetition of the adversary answer, `UK`, to the question `What is the largest country in the world?`. The Y-axis represent the probability of tokens, either the good answer, `Russia`, in bold line, or the wrong one, `UK`, in dashed line. Each color pair corresponds to a different LLM. The prompt used was different from the one initially proposed by [36].

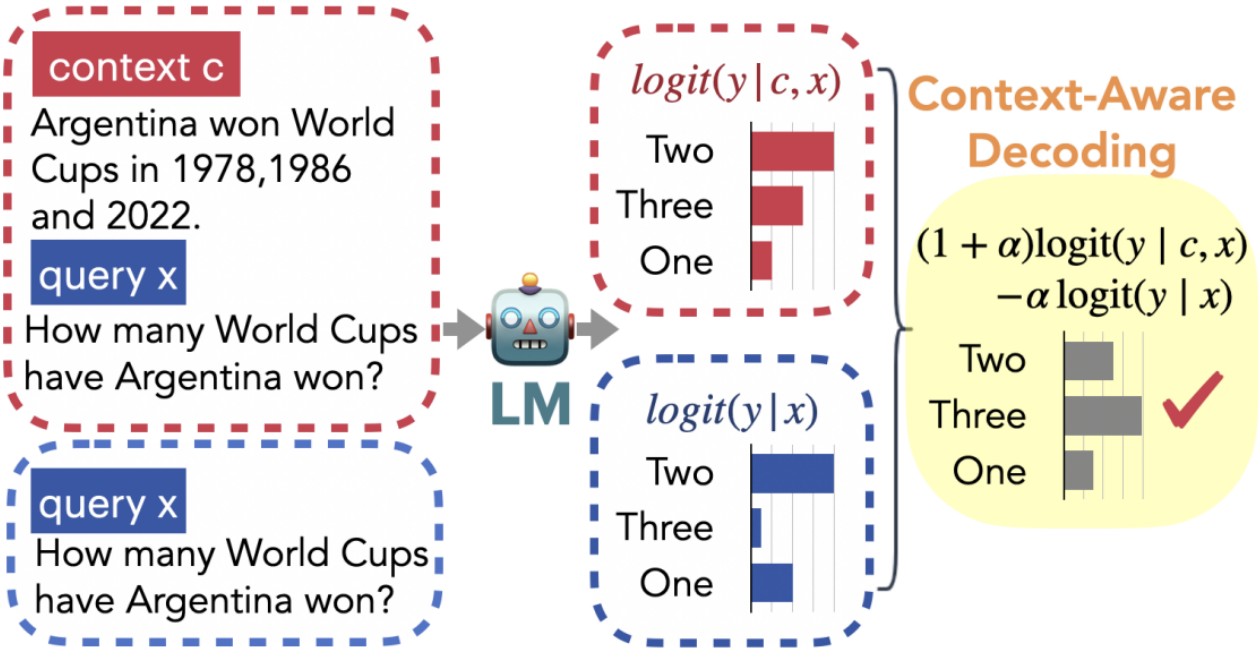


Figure 21: Representation of the functioning of the Context Aware Decoding that will compute the PMI between logits with or without a given context. Figure from [25]

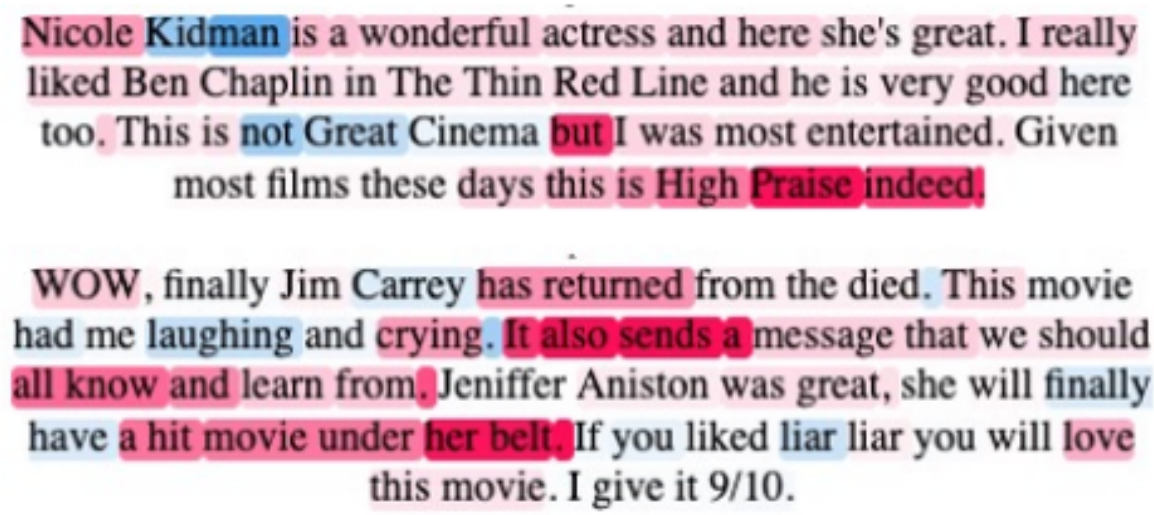
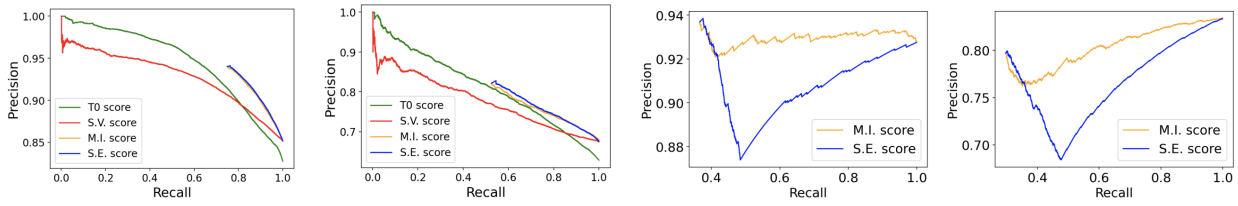


Figure 22: Representation of the Shapley values score for each of the token responsible for the D_{KL} in a sentiment analysis task, picture from [32]



(a) TriviaQA (b) AmbigQA (c) TriviaQA+WordNet (d) AmbigQA+WordNet

Figure 23: Results of [36] where they compare themselves to Semantic Entropy (from [18]) which yields similar results except on dataset with multi-label query (WordNet)[8]

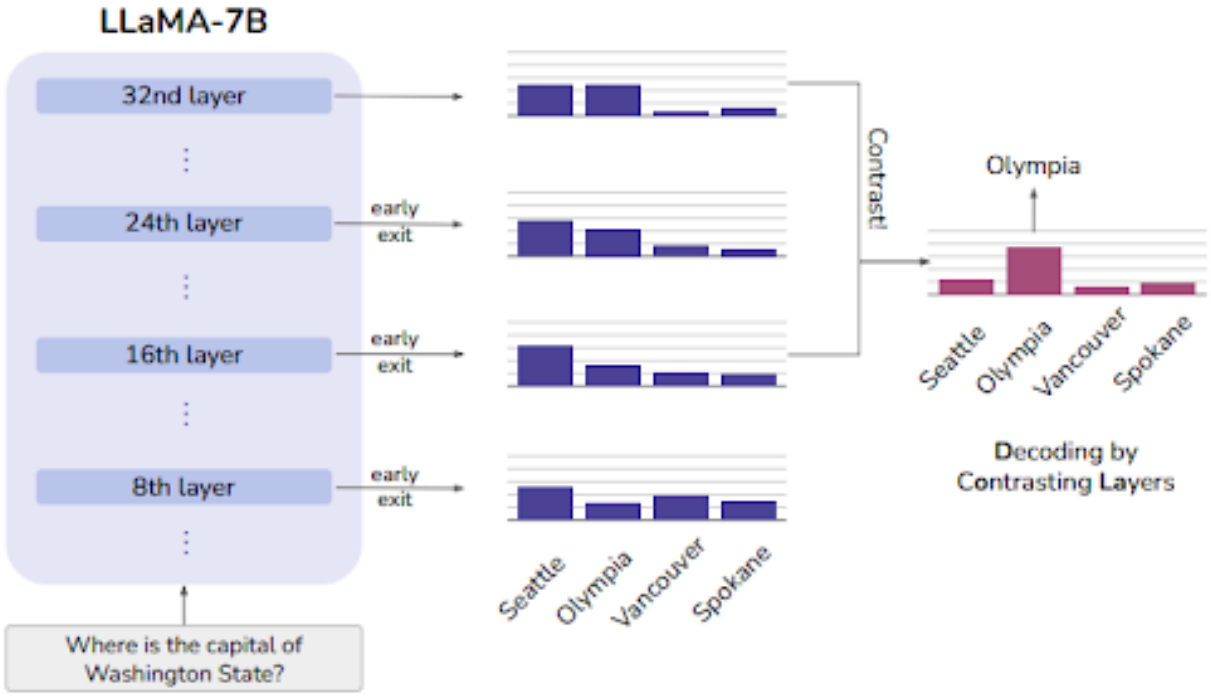


Figure 24: Representation of the DoLA framework that aims at looking for the most distant layers to the final one. This is therefore going one step further than just combining input and output. Figure from [4]

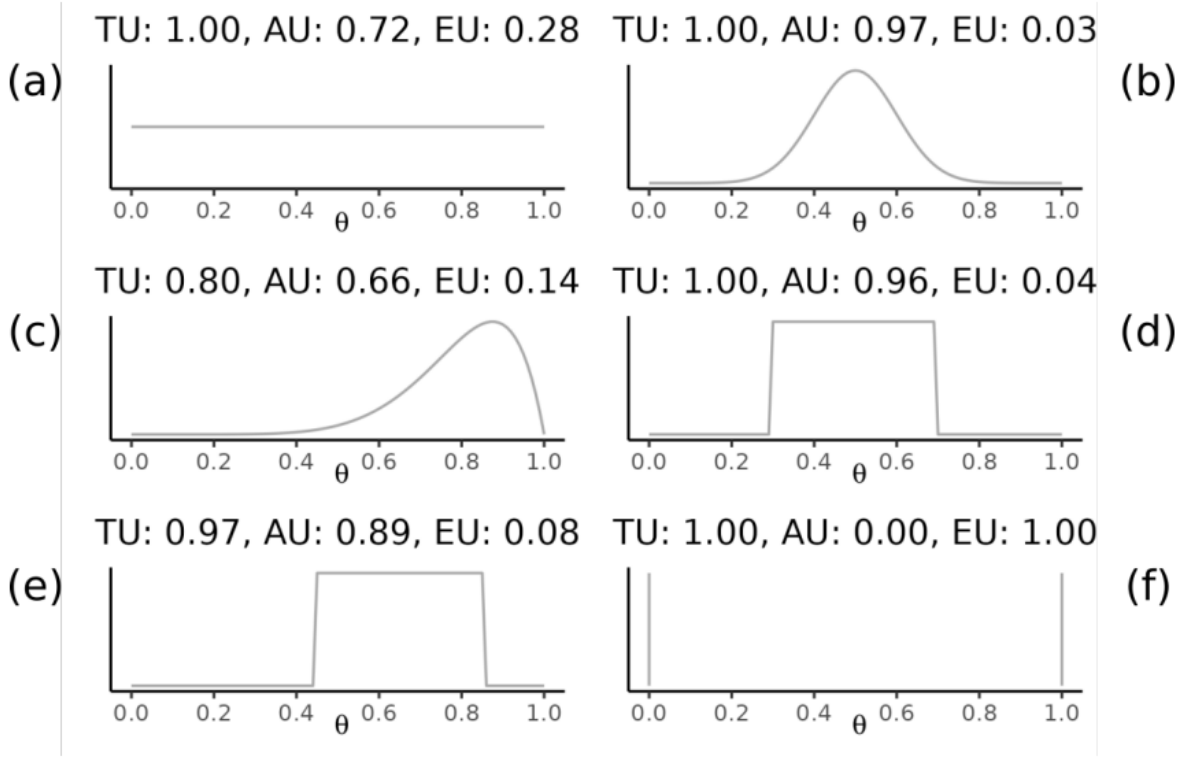


Figure 2: Different second-order distributions Q over the parameter θ of a Bernoulli distribution with associated uncertainty. (a) $\mathcal{U}[0, 1]$; (b) $\mathcal{N}(0.5, 0.01)$; (c) $\text{Beta}(8, 2)$; (d) $\mathcal{U}[0.3, 0.7]$; (e) $\mathcal{U}[0.45, 0.85]$; (f) $\frac{1}{2}\delta_0 + \frac{1}{2}\delta_1$.

Figure 25: Figure from [33] to depict the limits of the usage of the Mutual Information as a measure of lack of knowledge. It is more a measure of conflict as it can be depicted by a higher Mutual Information (E.U. for Epistemic Uncertainty) in panel (d) compared to panel (a) even if the distribution of the parameters of the models are more flat in (a) compared to (d).

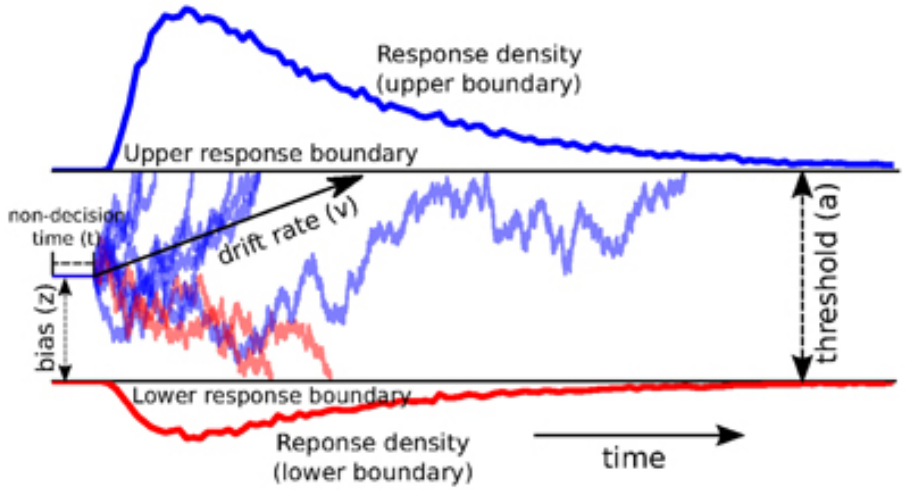


Figure 26: Illustration of the Drift Diffusion Model that illustrates how we might take our decision which is by accumulating knowledge before it reaches a certain threshold. The lack of knowledge is only useful before separating this is why the two etymologies of uncertainty *cerno* and *circinare* might have merged together in their usage.