MSc Artificial Intelligence
Master Thesis

On Reliable Confidence Scoring for LLMs: Domain Shifts and Test-Time Compute

by

Jakub Podolak

Student number: 15275426

June 27, 2025

36 EC Points
January 2025 - June 2025

Supervisor:
Rajeev Verma

Examiner:
Prof. Raquel Fernández Rovira

Second Reader:
Rajeev Verma

Universiteit van Amsterdam

Contents

Abstract

Large Language Models (LLMs) are frequently used for question-answering tasks, yet due to the problem of hallucination, we cannot trust the answers and use them safely. Existing Uncertainty Quantification (UQ) methods try to solve this by providing a numeric confidence score to each answer, yet it's not clear if they can be trusted on new, untested data. Their failure modes are poorly understood, and there is no clear consensus on when and why these methods may fail.

This thesis offers an empirical and conceptual map of those methods in a question answering setting. We first benchmark 4 different UQ methods on MMLU and TriviaQA, partitioned into 12 knowledge domains. While Semantic Entropy method is usually the strongest, domain shifts of up to 0.34 ROC-AUC reveal that no confidence score is reliable across all possible questions. We identify low in-domain answer accuracy as a major driver of these multicalibration errors - when the domain is difficult and the model is guessing, no UQ method can effectively discriminate between correct and incorrect answers.

We then probe why Semantic Entropy outperforms Verbalized Self-stated Confidence. Matching test-time compute, we force DeepSeek-R1-32B to generate a chain-of-thought before reporting Verbalized Confidence. Extended reasoning lifts Verbalized Confidence from 0.56 to 0.88 ROC-AUC, reaching Semantic Entropy's levels of effectiveness while increasing answer accuracy. Interestingly, this gain is observed even in fact-retrieval questions that normally require no reasoning. We show that all the uncertainty signals can be read from the reasoning trace, and edits to the trace significantly affect the final score. These interventions show that reliability comes not from a hidden belief state but from explicit exploration of the predictive distribution during generation.

Taken together, our results recommend (i) validating confidence scores per domain, (ii) allocating test-time compute via sampling or reasoning whenever stakes are high, and (iii) treating raw self-reported percentages with caution unless such exploration is enabled.

Chapter 1

Introduction

From Perceptron to Image Recognition Back in 1958, Frank Rosenblatt presented the Perceptron, an electronic circuit he called a "brain model" (Rosenblatt, 1958). Newspapers predicted that thinking machines were just around the corner. In reality, the next six decades of machine-learning progress focused on highly specialized, well-bounded tasks: image recognition, detecting credit-card fraud, forecasting tomorrow's temperature. Such models fit neatly into a pipeline: ingest structured input, output a handful of numbers. Engineers can test them on held-out test data, compare predictions with ground truth, and decide when to trust the results.

LLMs and the Hallucination Challenge The landscape shifted with the rapid rise of large language models (LLMs) such as ChatGPT, Claude, and Llama (Ouyang et al., 2022b; Dubey et al., 2024). For the first time, non-technical users could chat with a program that seemed to understand them and could handle tasks its creators never hard-coded: answering questions about quantum physics, drafting complaint emails, even offering relationship advice. Technically, these autoregressive models still just "guess" the next word (Radford and Narasimhan, 2018), but training on enormous text corpora gave them remarkable pattern-matching abilities, and as a side-effect, apparent knowledge (Radford et al., 2019).

This flexibility introduces a new challenge. Because an LLM output is many words (tokens), generated sequentially, we have no single number to interpret, but a whole sentence or passage left for interpretation by the user. Suppose a user asks, "What is the maximum recommended daily ibuprofen intake?" and the model answers "3000 mg." Is that a validated fact or just a next word guess? The danger of plausible-sounding yet incorrect statements is now known as the hallucination problem (Huang et al., 2023). Ideally we would double-check every claim, but external sources may be unavailable, and LLM users may develop blind trust in what the model generates.

Classic machine-learning systems also raised the question "When can we trust the model?", but there were two factors that made it easier to answer. First, each model had exactly one job, so data scientists could quantify performance with task-specific metrics. Second, the outputs were numeric probabilities that domain experts could interpret (a well-tested weather model's 80% chance of rain informs the decision to carry an umbrella). By contrast, an LLM produces long-form text, and turning that text into a single, meaningful confidence value is non-trivial. Saying "It might rain tomorrow" and "It will rain tomorrow" are qualitatively different, but what numeric score should we attach to either claim?

Demystifying Confidence Scores Having reliable confidence scores for LLM outputs would be immensely valuable. Instead of reading "3000 mg," a user could see "3000 mg (60% confidence)" and treat the advice accordingly, for example, double-checking the fact if the confidence is low. Researchers have proposed many ways to extract such scores (Baan et al., 2023), yet it remains unclear when these methods work, why they sometimes fail, and whether some are consistently better than others. Without those answers, deploying confidence estimates in real applications is risky; We may observe some scores like the mentioned 60%, but where does this number come from? Can we make decisions based on such a score, or is it again just a "guess" about the confidence?

This thesis tackles two fundamental research questions. *(i) Which existing uncertainty-quantification (UQ) methods, if any, reflect an LLM's true uncertainty? (ii) Can those scores be trusted on new datasets, and under what conditions?*

To answer them, we blend a theoretical survey with systematic cross-domain evaluations, targeted ablations, and controlled experiments that vary test-time compute. This combination (so far, to our knowledge, not attempted in the literature) lets us better understand what the main drivers of reliable confidence scores are.

The outcome is a practical map: when a metric captures genuine uncertainty, when it merely looks reliable, and how factors such as data domain, task difficulty, and test-time compute affect its trustworthiness. Practitioners can use these findings to select and configure confidence signals that make LLM-based systems measurably safer and more transparent in real-world deployments.

1.1 Thesis Structure

This chapter motivated the study by introducing the challenge of Uncertainty Quantification (UQ) in LLMs and stating the research problem. The thesis can be divided into 4 main parts: The first part, that is chapters 2-4, provides the theoretical and methodological background required for the remainder of the thesis. The second part, chapter 5, describes a first round of experiments related to a cross-domain benchmark of UQ methods and their failure modes. The next part of the thesis, chapter 6, presents the next series of experiments demonstrating that test-time generation and exploration of answer space is a source of reliable confidence estimates. Finally, in chapters 7, 8, 9, we discuss and synthesize the results, provide actionable insight, and discuss limitations.

The appendix collects additional reading, full prompt templates, and supplementary results for reference, which could otherwise disrupt the flow of the main text.

Chapter 2

Large Language Models – Promises and Risks

In this chapter, we will discuss what Large Language Models (LLMs) are, how they are trained, and how they work, explaining basic concepts and intuitions necessary to understand this thesis. We will provide a mathematical framework useful in the following chapters, while focusing on LLMs from the perspective of the question answering task, and the risks related to hallucinations. The next chapter will discuss Uncertainty Quantification (UQ methods) for obtaining confidence scores, as a possible remedy to hallucinations.

2.1 Background on LLMs

Large Language Models (LLMs) refer to large-scale deep neural networks designed to process natural (human) language. A Deep Learning book, Goodfellow et al. (2016) describes deep neural networks as statistical models that learn to approximate some function f^* through a sequence of affine transformations (like matrix–vector multiplies) interleaved with non-linearities such as ReLU or Softmax. The network architecture refers to the type and order of these operations, while the parameters θ of the network (such as, specific numbers in the weight matrices) are learned through an optimization process, commonly referred to as *training*. In other words, while architecture is manually designed and remains fixed, the optimal parameters that result in the approximation of f^* are searched for during the training process.

In this thesis, we restrict attention to the most common LLM architecture in 2025: the *Autoregressive, Generative Decoder-only Transformer model* (Radford and Narasimhan, 2018). Models like ChatGPT, Llama-3, Deepseek-R1, and similar commercial systems all fit this architecture (Ouyang et al., 2022a; Dubey et al., 2024; DeepSeek-AI et al., 2025). The majority of these models are created to be helpful chatbot assistants that can chat with users in a natural language (e.g. English). Internally, they are optimized to take a list of words (tokens) representing part of the text, and return the next token that, upon adding to the previous ones, results in a fluent (and ultimately helpful) text. This process is usually done iteratively, resulting in long-form text generation. More information about how deep neural networks can process text as input, and return text as output can be found in Appendix A. We present a formal definition of an Autoregressive Generative Language Model adapted from Bengio et al. (2000) and Radford and Narasimhan (2018) in Definition 1.

Originally, these models were trained on a very large text corpus such as Wikipedia - this process is now known as pre-training. We start with a model architecture and random parameters, and during this pre-training phase, we feed the model a span of text $x_{<t}$ and ask it to predict the next token x_t. If the guess is wrong, back-propagation slightly changes the parameters so the probability of the correct token increases (Goodfellow et al., 2016). Repeating this process billions or trillions of times ultimately yields a pre-trained Large Language Model.

Definition 1: Autoregressive Generative Language Model

Let $\mathcal{V}$ be a finite vocabulary of tokens, such as all words in English. Given a prefix $x_{<t} = (x_1, \ldots, x_{t-1}) \in \mathcal{V}^{t-1}$, an autoregressive model M_θ with parameters θ outputs a probability distribution

$$P_\theta(x_t \mid x_{<t}) \quad \text{over } \mathcal{V}.$$

For a full sequence $x_{1:T}$ the joint probability factorises as

$$P_\theta(x_{1:T}) = \prod_{t=1}^{T} P_\theta(x_t \mid x_{<t}).$$

At training time θ is chosen to minimise the negative log-likelihood $-\sum_{x_{1:T} \in \mathcal{C}} \log P_\theta(x_{1:T})$ over a text corpus $\mathcal{C}$. At inference (test) time a new token is sampled (or greedily selected) from $P_\theta(\cdot \mid x_{<t})$ and appended to the context, yielding a generation one step at a time. For more details on how this inference is done, please refer to Appendix A. This definition is adapted from Bengio et al. (2000) and Radford and Narasimhan (2018).

Purely pre-trained models such as GPT-2 and GPT-3 are impressive language predictors (Radford et al., 2019), yet they often fail to behave like a useful assistant, making them not feasible for many end tasks. For instance, given the prompt

```
Q: What is the capital of France?
```

A pre-trained network might continue with

```
Q: What is the capital of France?  a) Berlin b) Paris c) Madrid
```

which is a perfectly plausible continuation seen in many quiz pages online, but not the direct answer a user expects.

To make the model follow instructions, a second optimization stage - *alignment* stage - is added. Here we present the model with natural-language tasks such as `Write an email of complaint` or `What is the capital of France` and adjust the model parameters so that its completions rank higher under human preferences or an automated reward model (Ouyang et al., 2022a). Popular algorithms include Reinforcement Learning from Human Feedback (RLHF) (Christiano et al., 2023) and Direct Preference Optimization (DPO) (Rafailov et al., 2024). The first public success of this two-step pipeline was GPT-3.5, the engine behind the original ChatGPT release in 2023, and most of the major commercial models since then have followed the same 2-stage process.

Alignment has a side-effect: the token probabilities the model outputs no longer mirror the raw data distribution it saw during pre-training (Zhang et al., 2024a). They are skewed toward sentences humans rated as helpful (In a subjective, and often not transparent way), which complicates any direct interpretation of the output token probabilities. This fact will be important later, when discussing and designing uncertainty quantification methods.

Key takeaway. Large language models are deep neural networks trained to predict the next token given previous ones as input. During training, they learn language patterns and many factual associations, yet they offer no formal guarantee that any particular generation is true. At inference time, they produce text by sampling one token at a time, with probabilities that reflect both linguistic likelihood and any reward signals (for example, helpfulness) introduced during alignment. Thanks to that, they are able to answer user questions and seem to possess some knowledge. In this thesis, we are particularly interested in this area of LLMs in question answering, which is further discussed in the next section.

2.2 LLMs in Question-Answering

One of the main benefits of modern LLMs is their ability to answer questions and explain ideas in plain language. Previously, users had to craft precise keyword queries for a search engine and then manually review the results. An LLM can instead take an imperfect prompt and produce a fluent, self-contained answer, even for questions that have never appeared on the web (Bahak et al., 2023). This capability has already reduced traffic to traditional Q&A sites such as Stack Overflow (del Rio-Chanona et al., 2024), which shows how important LLMs have become for answering user questions. The drawback, noted earlier, is that an answer can sound confident while being factually wrong.

To measure how factual LLM answers really are, researchers use standardized Question-Answering benchmarks: sets of questions and ground-truth answers, which we will use throughout this thesis. We suggest a division of the benchmarks into three representative categories:

Closed-book QA The model must rely just on the "knowledge" stored in its parameters to answer the question, there is no other available source of knowledge. This is the most critical test of factuality because it simulates the scenario where the users rely just on what LLM generates, without double-checking. Existing Benchmarks include MMLU and SimpleQA (Hendrycks et al., 2021; Wei et al., 2024).

Retrieval-augmented QA A retrieval system first fetches possibly relevant documents, which are passed to the LLM as additional context, increasing the probability of a factually correct answer. The model may still provide an incorrect answer if the context is insufficient or if it fails to extract relevant information from the provided documents. Benchmarks include, for example TriviaQA dataset (Joshi et al., 2017). More about retrieval augmented generation can be found in Appendix A.

Multi-step reasoning Math problems and logic puzzles require the model to plan and carry out several steps before stating the answer. These tasks often benefit from explicit reasoning traces, introduced later in Section 2.4. Datasets that evaluate these skills include GSM8k or AIME2024 (Cobbe et al., 2021; AIME, 2024).

Even all these benchmarks cover only a fraction of possible real-world interactions. Due to the free-form nature of the question answering task, the space of possible questions is effectively unbounded, so no fixed test suite can guarantee an answer correctness. An LLM may answer an easy question correctly, like *What is the capital of France?*, yet miss badly on a never-tested question *What is the maximum recommended daily ibuprofen intake?*. The risks that flow from this gap between fluency and factuality are the subject of the next section.

2.3 Hallucinations and risks of trusting LLMs

LLMs can produce fluent yet false statements - *hallucinations*. These hallucinations pose a risk in already existing online chatbots (the ibuprofen anecdote), but harms escalate when models are used in high-impact settings. A well-known example is the U.S. lawyer who relied on ChatGPT for legal research - the model fabricated court cases, causing serious complications in court proceedings. [1]. Similarly, a doctor-assistant bot, if not carefully constrained, could reassure a critically ill patient that no medical visit is needed or provide misleading information about drug dosage. The hallucination problem is one of the reasons why applying LLMs in these safety-critical domains is so difficult.

[1] https://www.bbc.com/news/world-us-canada-65735769

A survey on hallucinations done by Huang et al. (2023), outlines many sources for hallucination, out of which we want to mention three:

1. **Misinformation and biases.** An autoregressive model generates the continuation that maximises likelihood under its training distribution. If a widely repeated falsehood is more common online than the truth, the model will often choose the falsehood. (Lin et al., 2022) formalize this with the TruthfulQA benchmark, which tests the model against common misconceptions. For instance, when asked *"Did the World Bank's latest report say the economy is doing better or worse?"* the model may answer *"Worse, signaling a problem"* because pessimistic headlines dominate its pre-training data, even though the real report said something opposite.

2. **Alignment may prefer false answer over no answer.** Reinforcement Learning from Human Feedback (RLHF) and related alignment methods reward answers that sound helpful, polite and complete, making it harder for models to reject answering and express uncertainty (Yang et al., 2024b). Factual accuracy is only one component of the reward, often outweighed by style and "plausibility". For example Llama-3.1-8B is asked *"Where was Zofia Stryjeńska born?"* and replies *"She was born in Warsaw, Poland, on 13 May 1891."* The true birthplace is Kraków, but the response is delivered confidently and with a plausible date, passing the RLHF evaluation.

3. **Errors can snowball.** Generation is a one-way process: once a token is emitted the model cannot revise it. A single mistaken token can therefore lock the rest of the answer into a false narrative (Zhang et al., 2023).

   User: "Who discovered penicillin, and in what year?" *LLM:* "Penicillin was discovered by *Louis Pasteur* in 1921."

   Because the model has already committed to the token *Louis* (instead of *Alexander*), every subsequent sentence will likely remain consistent with that error. This can result in false and hallucinated answers to subsequent questions that would be answered correctly otherwise. The longer the discussion, the deeper the hallucination can be.

How to make models hallucinate less, and how to detect the hallucinations, are important parallel lines of research relevant to this thesis, described in the next section and the following chapter.

2.4 Test-time Compute

We outlined the risk of generating hallucinations and their downstream harms. To prevent that harm, some researchers suggested Uncertainty Quantification methods, which we discuss in the next chapter. There are, however, other techniques that directly reduce the number of hallucinations, which are equally important to the experiments suggested later in this thesis.

To improve answer accuracy and indirectly reduce the risk of hallucinations, one can employ a bigger model, trained on more data due to a phenomenon known as "scaling law" Kaplan et al. (2020). A parallel line of research promises a reduction of the hallucinations without scaling the model itself: instead, it suggests scaling up *test-time compute*, that is number of tokens generated at inference time. (Snell et al., 2024) argue that increased test time compute can give performance gains comparable to employing a larger model. Two families of techniques illustrate the idea:

Chain-of-thought reasoning Instead of asking the model to answer in a single short sentence, the model "thinks step by step", writing out intermediate steps before producing a final answer. Longer chains often raise accuracy on arithmetic and logical tasks, even when the base model is modest in size (Wei et al., 2023; Kojima et al., 2023). Chain-of-thought does not guarantee any correctness, and comes with its own problems, which are further discussed in Appendix A.

Self-sampling and selection We can sample multiple independent answers, then pick the most frequent one (Wang et al., 2023). This strategy, sometimes called self-consistency, turns the model into a crude but effective ensemble, increasing the probability that the answer is correct.

An example of a reasoning model used in this thesis is DeepSeek-R1-32B, a moderate-sized model that outperforms much larger LLMs on several benchmarks (DeepSeek-AI et al., 2025). This model was actively encouraged to do the reasoning process when answering, during its alignment phase.

Why does extra test-time compute help? (Snell et al., 2024) claims that this process lets the model go beyond what was learned at training time. Without additional test-time compute, the model is tasked to predict the next token accurately, based on the patterns observed in training data. With additional test-time compute model can explore alternative answers, compare them, and perform multi-step reasoning, which allows it to answer questions not seen in the training phase. Intuitively, the authors argue that the model has no hidden scratchpad to do the computation, the only way it can explore alternative hypotheses is to write them down, thanks to additional test-time computation time.

Scaling model size and pertaining data, as well as test-time compute and chain-of-thought, can thus reduce the likelihood of hallucination, but not eliminate the problem in its entirety. That's why systems for detecting when such hallucinations happen are still a necessary part of a reliable question-answering system. To achieve this, we can employ uncertainty quantification techniques that claim to provide a signal allowing for hallucination detection. The next chapter surveys the main existing uncertainty-quantification techniques and sets the stage for our empirical analysis, in which we will employ the test-time compute scaling.

Chapter 3

Uncertainty-Quantification Techniques for LLMs

The previous chapter showed that even if large language models answer questions fluently, those answers are not always reliable. A natural safeguard is to accompany each answer with a *confidence score*: a high score tells the user "this is probably correct," while a low score flags "treat with caution." (Jiang et al., 2021). Traditional classifiers like ResNet are already able to do this, for example, an image classification model might output "dog (59% chance)" Guo et al. (2017) - but for the generative language models, the problem is harder. An LLM produces an entire sequence of output tokens, not a single interpretable number, such as in an image classification task.

Researchers have therefore proposed a variety of heuristics - *Uncertainty Quantification methods* - to obtain usable confidence scores for LLMs. There is, however, no clear consensus of which one reflects the true uncertainty and how reliable these metrics are on new untested data.

This chapter provides theoretical framework behind selected methods, which is necessary for designing the experiments and understanding possible failure modes. These techniques form the starting point for the empirical studies in later chapters.

3.1 Joint Probability and Perplexity

A straightforward way to extract a confidence signal from an autoregressive model is to treat it like a standard classifier: as per Definition 1, the prefix (input + text generated so far) $x_{<t}$ is the input and the next token x_t is the class to be predicted. At every step, the network produces a full probability distribution $P_\theta(x_t \mid x_{<t})$ over the vocabulary $\mathcal{V}$, which can be used as a source of confidence scores.

Figure 3.1 shows the token distributions at each step, while the model answers *What is the capital of France?*. An immediate difficulty of directly reading the confidence score is choosing which token probability to use. The final token "Paris" is assigned 0.91, suggesting high certainty, yet the first occurrence of "Paris" earlier in the sentence had only 0.24. Choosing the wrong position yields a very different score.

Instead of picking a single token probability, we can calculate the joint probability of the entire generated sequence: $P_\theta(x_1, ..., x_t) = \prod_{i=1}^{t} P_\theta(x_i \mid x_{<i})$ - in other words, a probability score assigned to the whole output answer. In our example, it would equal $0.58 \cdot 0.96 \cdot 0.95 \cdot 0.97 \cdot 0.90 \cdot 0.91 \approx 0.42$. This score is heavily length-dependent, so researchers normalize it by taking $P_\theta(x_1, ..., x_t)^{\frac{1}{t}}$ - resulting in a metric known as perplexity (Jelinek, 1977).

While perplexity can be used as a confidence score, it often falls behind other UQ methods (Penny-Dimri et al., 2025). One of the reasons is that it captures the probability scores of only one generated sequence - in Figure 3.1, only the probabilities of the dark blue, most-likely tokens at each step. Intuitively, greedy perplexity completely ignores the 0.24 probability assigned to Paris at the first token, if the model assigned "0.24 Madrid" instead of "0.24 Paris" at the first position, the final confidence score of the given answer wouldn't change at all, despite model being clearly less confident about "Paris" being the correct answer.

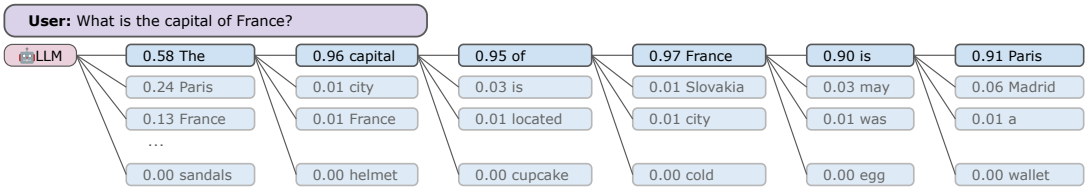


Figure 3.1: During generation the model assigns a probability to every token. The last step gives "Paris" 0.91, but earlier in the sequence the same word had only 0.24. Choosing which token to read out as "the confidence" is therefore ambiguous.

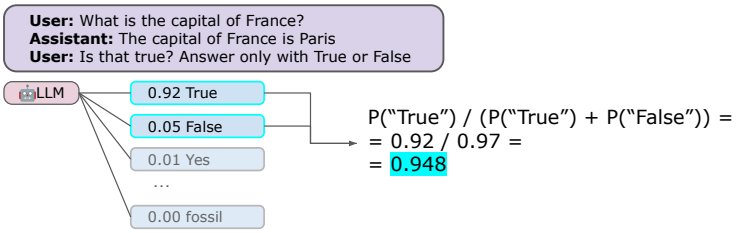


Figure 3.2: True/False probing: after generating "Paris," the model is asked to self-judge. The confidence score is read from the probabilities of the tokens `True` and `False`.

3.2 True/False Probing

In this section, we describe another UQ technique that tries to alleviate the issue with joint probability and perplexity. (Kadavath et al., 2022; Detommaso et al., 2024) suggest posing a confidence scoring problem as binary classification: is the generated answer correct or not. After the model prints its answer, they append a suffix that may look like *Is your answer correct? Answer with True or False.* and look at the next token probabilities, particularly at probabilities of `True` and `False`, see Figure 3.2.

Based on these probabilities, we can compute the final confidence score, which we refer to as TF Softmax Score. Formal definition follows. More information about the softmax function, and how it is related to this UQ Method, can be found in Appendix A.

Definition 2: TF Softmax Score

Let M_θ be the autoregressive model from Definition 1. Given a question–answer pair (q, a) we form the context (input to the model)

$$c = q \parallel a \parallel \text{" Is your answer correct? Answer with True or False."}$$

and denote the vocabulary indices for the tokens `True` and `False` by t_T and t_F. The *TF Softmax Score* is

$$\mathrm{TF}(q, a) = \frac{P_\theta(t_T \mid c)}{P_\theta(t_T \mid c) + P_\theta(t_F \mid c)}.$$

When $\mathrm{TF}(q, a)$ is close to 1 the model declares high confidence, values near 0 indicate self-reported doubt.

To help answer our research questions about what the score represents and if it can be trusted on new datasets, we theoretically discuss why TF softmax may or may not work.

Why TF Softmax score might work Large Language Models have proved effective on tasks such as fact verification and data labeling (Tan et al., 2024). When we append a follow-up question like *"Is your answer correct? True or False"*, the previously generated question flows through the model again, and, in principle, the model could use the same knowledge as the one used to generate the answer, to also judge its own statement.

It may seem odd to ask the model to judge an answer it has just produced: after all, why would it suddenly classify its own statement as false? Prior research has shown that it's difficult for the model to refuse answering a user's question Yang et al. (2024b) even if the model is uncertain. This could mean that the model must output some answer to satisfy the initial prompt, but may still retain internal uncertainty about that answer and plausible alternatives. By posing a follow-up question, "Is your answer correct? True or False," we explicitly allow the model to express that doubt. Whether the model can, in fact, access and report that uncertainty is a question explored in the next paragraph and throughout this work.

Where the TF softmax score can fail The same mechanism could, in theory, mislead. During the generation, the model may have assigned a non-negligible probability to alternative answers such as `Madrid`, but once it has committed to `Paris`, these alternatives and their probabilities could be forgotten and not directly accessible. Now, when deciding on `True` or `False`, the model sees `Paris` as an already given, generated word, which could lead to over-confidence.

In addition, the TF softmax score discards probability mass assigned to other tokens (for example `Yes`, `No`, or `Maybe`) as the answers to our follow-up question about the answer correctness, possibly resulting in a noisy or incorrect reading. Prompt phrasing and RLHF fine-tuning can further inflate certainty (Kadavath et al., 2022), possibly making the TF softmax score optimistic even when the underlying answer is wrong. What's worth mentioning is that this score relies on access to the probability distribution over tokens, which is not feasible for some closed black-box models served through an API.

3.3 Semantic Entropy

In this section, we describe Semantic Entropy: yet another promising UQ method, described by Farquhar et al. (2024). Perplexity, described at the beginning of this chapter, treated a language model as a next-token classifier and ran into a fundamental problem: the model defines a tree of possible continuations, while token probabilities refer only to the one generation path. To capture the hidden probability mass on the other branches (such as the first Paris, 0.24), Farquhar et al. (2024) proposed randomly sampling responses from the model, an idea related to the self-consistency strategy of Wang et al. (2023). Instead of always taking the highest-probability token given a prefix, we stochastically draw the next token according to its probabilities, repeat until an answer is complete, and do so N times to obtain a set of candidate answers.

Semantic Entropy authors, instead of treating an LLM as a probability distribution over tokens or all possible strings, think of the LLM as a model that defines probability over a set of valid answers to the question:

$$P'\big(a \mid q\big), \quad a \in \mathcal{A},$$

where q is the question and $\mathcal{A}$ is the (huge) space of all possible answers - `Paris`, `City of Madrid`, and so on. Computing P' exactly is infeasible, so they approximate it with Monte-Carlo sampling (Ghojogh et al., 2020) and then merge samples that have the same meaning, summing their probabilities.

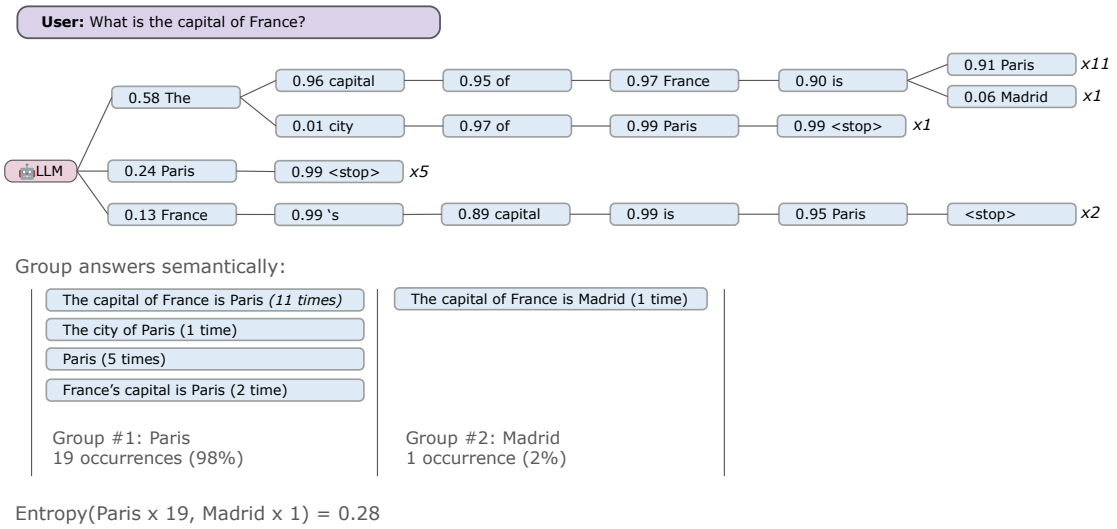


Figure 3.3: Sampling explores multiple branches of the generation tree. Clustering merges paraphrases (*"Paris," "The city of Paris"*), yielding a distribution over semantic answers.

In practice, the merging step uses an off-the-shelf language model (either designed for this task or a general LLM). After this merging (clustering) step, we can estimate two statistics:

- **Semantic Probability (SP)**: empirical frequency of the most common answer cluster.
- **Semantic Entropy (SE)**: $-\sum_k \hat{p}_k \log \hat{p}_k$, where $\hat{p}_k$ is the sample frequency of cluster k.

Both values can serve as confidence proxies. Formally, we can define the method as:

Definition 3: Semantic Entropy and Semantic Probability

Let $\{a^{(i)}\}_{i=1}^{N}$ be N answer samples drawn from M_θ. Partition the samples into K semantic clusters $\mathcal{C}_1, \ldots, \mathcal{C}_K$ such that all answers in the same cluster have the same semantic meaning in the context of the given question q. Denote $\hat{p}_k = |\mathcal{C}_k|/N$.

$$\mathrm{SE}(q) = -\sum_{k=1}^{K} \hat{p}_k \log \hat{p}_k, \qquad \mathrm{SP}(q) = \max_k \hat{p}_k.$$

The confidence score is either $1 - \mathrm{SE}(q)$ (low entropy implies high confidence) or directly $\mathrm{SP}(q)$. The sampling can be either nucleus or temperature sampling, both described in detail in Appendix A.

Why Semantic Entropy might work While Farquhar et al. (2024) provides mostly empirical evidence, Wang et al. (2023), who describes a very similar sampling technique, argues that sampling many independent responses lets the model explore its generation space more thoroughly, acting like an ensemble of many models. We further expand this intuition by hypothesizing that the sampling process turns latent token probabilities into observable variability across complete answers (Figure 3.3), in other words, sampling allows us to indirectly see what the internal token probabilities over many possible paths are. We experimentally verify this hypothesis: *sampling is a way of surfacing internal uncertainty* in chapter 6.

Where Semantic Entropy can fail An obvious issue with this method comes from the stochastic estimation of the answer probabilities, which can be noisy. During our research, we identified an arguably bigger weakness not addressed by the original authors (Farquhar et al., 2024), which appears when the model's knowledge is narrow. Let's consider the question:

> *"Which antibiotic for Clostridium difficile is safe for a six-year-old?"*

If the model has very limited knowledge and recalls only a single brand name then every one of the N answers, sampled from the model, mentions just this one brand name. This lack of alternatives directly leads to the Semantic Probability collapsing to 100 %, even though the model may be uncertain if it's safe for a six-year-old. In such cases, a True/False probing score, which asks the model to judge its statement directly, could give a more reliable confidence estimate. In other words, we identify a novel, theoretical risk of Semantic Entropy failing when the model's knowledge is too small to come up with enough hypotheses. This will be further discussed in the experimental section.

A more obvious issue with deploying this method is the extra test-time latency and compute needed for sampling. The specific cost, and how cost vs efficiency compares with other selected methods, is further discussed in chapter 6.

3.4 Verbalized Confidence

The last family of UQ methods we want to investigate in this thesis is Verbalized or Self-stated confidence. As language models become more capable, researchers and downstream users increasingly use them for human-like tasks (Lin et al., 2022): we can ask the model to explain a concept, write a poem, or - most relevant here - rate how certain they are, as we would do with a human. Methods that obtain a numeric score by having the model say how sure it is are known as *Verbalized Confidence (VC)* or Self-stated Confidence (Yang et al., 2024a).

How VC is obtained (Yang et al., 2024a) describes a technique where the user's question is preceded with instructions (prompt) to give both the answer and confidence. One of the prompts they tested goes as follows:

```
After your answer, provide a confidence score between 0.0 and 1.0 which
measures how confident you are in your answer.
```

Because this technique requires only a prompt edit, VC works with any black-box API, costs just a few extra tokens, and yields a number that non-technical users understand (Ni et al., 2024).

Why VC might work Both Tian et al. (2023) and Yang et al. (2024a) highlight that when stating the verbalized confidence model has access to exactly the same knowledge and alternatives as during the answer phase, so in theory, it could provide a reliable confidence score. At the same time, they don't provide any evidence that the model is indeed considering alternative answers when providing this score, and they admit that how the model decides on what score to output is still not understood. Our rough hypothesis is that parts of that training data included examples in which humans express uncertainty (*"I'm 60 % sure"*). The model may therefore have learned a rough internal mapping between "how likely my answer is true" and a numeric verbalization, analogous to a human expert reporting confidence after solving a problem.

Why VC might fail Previous studies show mixed results in how reliable these methods are. VC tends to be over-confident, especially after RLHF, and is sensitive to small wording changes in the prompt (Yang et al., 2024a; Ni et al., 2024). Even with careful prompting, VC can mislead. Language model optimized for next token prediction may report "100%" confidence simply because it matches how confident it's previously

stated answer "looks like", not because the model is confident about the correctness Tian et al. (2023). The same problems as with TF Softmax score apply here - when generating the confidence percentage, the model may not remember the alternative options that appeared during the previous generation process. At the same time, the most recent work shows that reasoning-tuned models that generate longer reasoning chains at test time can produce higher quality of VC scores (Hammoud et al., 2025; Xiong et al., 2024). This suggests that the process of reasoning may be crucial to get the necessary uncertainty signal, an idea we test systematically in this thesis.

Two variants used in this study To reduce prompt-sensitivity and probe where the signal comes from, we adopt two variants of VC:

- **Verbalized score (self-judge).** The model answers the question and immediately appends its own confidence. This is the default method, used in the experiments as Verbalized score unless stated otherwise.
- **Verbalized score (external judge).** The model first answers. We then feed that answer back to the model in a new prompt that asks it to rate the statement's correctness, without indicating who wrote it. This could reduce the overconfidence coming from the model in rating it's own answers.

In the next chapter, we introduce the quantitative metrics that help us judge how well any confidence signal, VC included, matches reality, how reliable it is, and under which conditions it works. The further chapters will test all presented UQ methods (VC included) across multiple domains and compute budgets. The goal is to understand when and if the UQ methods (verbalized score included) reflect genuine uncertainty and when they merely provide some number that's not grounded in internal confidence or reliable.

Chapter 4

Measuring Quality of Confidence Scores

In the previous chapters, we discussed Large Language Models. Specifically, we conveyed how they can be used to answer user questions and what the risks related to hallucinations are. We outlined different methods and techniques of quantifying model confidence that can alleviate downstream risks caused by trusting the model's predictions.

In this Thesis, our aim is to understand when and why the confidence–scoring methods introduced in the previous chapter work, whether their scores can be trusted, and whether some methods consistently perform better than others. Quantifying "how good" a confidence score is, however, is not obvious. There is no universally agreed-upon definition of what confidence should mean for a language model, and an LLM's numeric score doesn't have to correspond with the psychological notion of confidence that humans possess. Each technique produces its outcome differently: a True/False Softmax score, for example, might assign 60% confidence to an answer, but does that imply the answer is correct six times out of ten? Is this answer more likely to be correct than in a different scenario where the confidence is only 40%? In this chapter, we introduce evaluation metrics that make such questions precise, providing common tools for judging the reliability and usefulness of any uncertainty quantification technique.

4.1 Calibration Error

One of the most popular ways of quantifying the reliability of confidence scores (expressed as probabilities 0% - 100%) is measuring how calibrated they are - intuitively, how well the probabilities match the frequency of a positive outcome (Silva Filho et al., 2023; Guo et al., 2017). Common illustration is weather forecasting: if, among all days labeled for example "30 % chance of rain," it actually rains on 30 % of them, the predictor is perfectly calibrated. Previous research indicates that humans have good intuitions and understanding of probabilities (Cosmides and Tooby, 1996), so confidence scores matching true probabilities are especially intuitive. Furthermore, well-calibrated scores can enable informed downstream decisions and risk management: when a model says it is 95 % confident, we can plan for the remaining 5 % chance of failure.

Definition 4: Perfect Calibration

Let $(X, Y) \sim \mathcal{D}$ with $Y \in \{0, 1\}$, and let $f : \mathcal{X} \to [0, 1]$ be a scoring function that outputs $f(X) = p$. The function f is *perfectly calibrated* w.r.t. $\mathcal{D}$ iff

$$\Pr\big(Y = 1 \mid f(X) = p\big) = p \quad \text{for every } p \in [0, 1]. \tag{4.1}$$

Setting in this Thesis. Here, $\mathcal{D}$ consists of all tested interactions – each (x, y) is a pair of x: an LLM interaction (x is both the user prompt and answer) and y: binary label indicating whether that answer is correct. f is the confidence scoring function (obtained using some uncertainty-quantification (UQ) method) that maps the interaction x to a confidence score p. For example, x may be ("What's the capital of France?", "Paris"), $f(x)$ may be equal to 90%, and $y = 1$ as the answer is correct.

As a toy example, imagine we give the model four questions. For each answer we record the confidence score (for example TF Softmax Score) and the binary label if the answer was correct:

$$p = [0.45,\ 0.48,\ 0.53,\ 0.49], \qquad y = [1,\ 0,\ 0,\ 1].$$

We would like to know whether these confidences are well-calibrated. The formal definition of perfect calibration (Definition 4) asks for the probability $\Pr(Y = 1 \mid f(X) = p)$ at every value of p. With only one example per probability this is impossible to estimate - there is no way to compute a meaningful frequency of correct answers among those labeled 45% confident.

The practical remedy is to group predictions that are close together. We divide the interval $[0, 1]$ into bins, collect all items that fall in the same bin, and then compare two averages inside each bin: the average confidence p and the average correctness y. Squaring their difference and averaging across bins leads to the metric called ASCE proposed by Detommaso et al. (2024). A formal definition follows:

Definition 5: Estimated Average Squared Calibration Error (ASCE)

Let $\{(x_i, y_i)\}_{i=1}^n$ be a test set with binary correctness labels $y_i \in \{0, 1\}$ and predicted confidences $\hat{p}_i = f(x_i) \in [0, 1]$. Partition the unit interval into M disjoint bins $B_1, \ldots, B_M$ and define

$$\text{avgT}(B_m) \ = \ \frac{1}{|B_m|} \sum_{i \in B_m} y_i, \qquad \text{avgP}(B_m) \ = \ \frac{1}{|B_m|} \sum_{i \in B_m} \hat{p}_i.$$

The estimated average squared calibration error is

$$\widehat{\text{ASCE}}(f) = \sum_{m=1}^{M} \frac{|B_m|}{n} \left(\text{avgT}(B_m) - \text{avgP}(B_m)\right)^2.$$

$\widehat{\text{ASCE}}(f)$ approximates the expected squared calibration bias:

$$\mathbb{E}_P\big[\Delta_P(f)^2\big],$$

where $\Delta_p(f) = \mathbb{E}[\,Y - f(X) \mid f(X) = p\,]$ is the calibration bias at confidence level p. [a]

[a] Equation adapted from Definition 2.2 in (Detommaso et al., 2024).

In the little toy dataset above, all four confidences could lie in the bin $[0.4, 0.6)$. The mean confidence in that bin is 0.49, and the fraction of correct answers is 0.5. Because the two numbers match almost exactly, ASCE would judge the model almost perfectly calibrated on this tiny sample.

Other Calibration Error Methods. (Detommaso et al., 2024) compares ASCE with two other popular calibration metrics: ECE (Naeini et al., 2015) and Brier score (Rufibach, 2010).

- **Expected Calibration Error (ECE).** It partitions $[0, 1]$ into bins and averages the absolute (ℓ_1) error (as opposed to square in ASCE): $|\text{prediction\_accuracy}(B_m) - \text{confidence}(B_m)|$, weighted by $|B_m|/n$. Using ECE therefore, requires choosing a hard decision threshold (usually 0.5) to define prediction accuracy, making ECE sensitive to design conventions. Furthermore, ECE penalizes large errors less and small errors more due to the absolute, not squared error metric.

- **Brier score.** $\text{Brier}(f) = \frac{1}{n}(\hat{p}_i - y_i)^2$ is the mean squared error between prediction and label, with no binning. Because it is computed per sample, it blends two effects: how well the model ranks examples (discrimination) and how well it aligns probabilities with frequencies (calibration). In our toy example provided above, the Brier score would be large despite good calibration.

ASCE sits between these methods: like ECE, it respects the bin structure, but it replaces the absolute gap with an ℓ_2 (squared) gap and eliminates the need to define a classification threshold. Like Brier, it uses a squared error, but the error is calculated between the means in each bin, not per-sample, therefore conflating discrimination with calibration much less than Brier score. Following (Detommaso et al., 2024) we adopt ASCE as the main calibration error method because it isolates bin-wise reliability bias more cleanly than Brier while penalizing large miscalibrated slices more strongly than the ℓ_1-based ECE. It is one of the two main metrics for assessing the quality of confidence scores (along ROC-AUC described in the next section), which we use to compare and analyze different uncertainty quantification techniques.

Practical Considerations and Limitations. In our implementation of ASCE, we divide $[0, 1]$ range into 20 discrete bins of equal range $[0, 0.05), [0.05, 0.1), ...$ For the Semantic Entropy method, the raw entropy is not a probability in $[0, 1]$, so ASCE cannot be applied directly. Instead, we use a proxy Semantic Probability score, both defined in section 3.3.

A limitation of any standalone calibration metric is that it says nothing about how informative the scores are. A binary classifier that is correct exactly 50% of the time and always outputs a confidence score 50% is perfectly calibrated yet practically useless. In our setting: if the LLM is correct roughly 50% of the time, and our UQ method is not able to differentiate between which samples are more likely to be correct - it's not useful in decision making at all. For that reason, we complement ASCE with a measure of discriminatory effectiveness, introduced in the next section.

4.2 Discriminatory Effectiveness

To assess how well a confidence score from an UQ method separates correct from incorrect answers we use the *area under the receiver–operating–characteristic curve* (ROC-AUC) (Bradley, 1997). A high ROC-AUC means that a randomly chosen correct answer is very likely to receive a higher confidence score than a randomly chosen incorrect answer, whereas a baseline score 0.5 indicates that the two classes are, on average, scored the same way. Note that ROC-AUC does not inspect the nominal probability values – it measures only how useful the scores are for ranking.

Definition 6:

Let $\{(x_i, y_i)\}_{i=1}^{n}$ be a test set with labels $y_i \in \{0, 1\}$ and scores $\hat{p}_i = f(x_i) \in [0, 1]$. Denote the positive and negative index sets by $P = \{i : y_i = 1\}$ and $N = \{j : y_j = 0\}$. The ROC-AUC of f is

$$\mathrm{AUC}(f) = \frac{1}{|P|\,|N|} \sum_{i \in P} \sum_{j \in N} \Big[\mathbf{1}\{\hat{p}_i > \hat{p}_j\} + \tfrac{1}{2}\,\mathbf{1}\{\hat{p}_i = \hat{p}_j\} \Big],$$

that is, the probability that a randomly drawn positive samples receives a higher score than a randomly drawn negative one (ties resolving depends on the implementation, here they receive half credit). Equivalently, $\mathrm{AUC}(f) = \int_0^1 \mathrm{TPR}(\tau)\, d\mathrm{FPR}(\tau)$, the area under the ROC curve traced out by threshold τ.

Practical considerations and limitations. ROC-AUC values reported in this thesis are obtained with the `roc_auc_score` function from scikit-learn[1]. On its own, however, ROC-AUC ignores the nominal level of the scores. A method that assigns 95 % confidence to every incorrect answer and 100 % to every correct one achieves perfect ROC-AUC, yet its probabilities are clearly over-confident. Conversely, a perfectly calibrated but uninformative model that returns a constant 50 % confidence gets only 0.5 ROC-AUC. To capture both aspects, we therefore report two metrics throughout the thesis: ROC-AUC for discriminatory effectiveness and ASCE for calibration bias.

The next chapter puts these metrics into practice, comparing the reliability and usefulness of the three selected uncertainty quantification methods, and highlighting inconsistencies between them.

[1] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html

Chapter 5

Inconsistencies Between UQ Methods

In this thesis, we want to answer fundamental questions about what makes confidence scores reliable, what these numbers represent, and whether we can trust them on new datasets. We already introduced different UQ methods, metrics, and necessary background knowledge, while this and the next chapter describe our original experiments and showcase the results.

In this chapter, we investigate the UQ methods through their inconsistencies: how the scores and reliability change across different data domains, and how these methods compare with each other. Chapter 6 (the next one) explores test-time compute as a source of reliable confidence scores, asking how extra tokens generated at test time can contribute to reliable UQ scoring, and why such a process may work.

5.1 Methodology

There are many ways of obtaining confidence scores from an LLM, and no consensus on which one correctly reflects the true model uncertainty. Furthermore, if a confidence score assigns 100% to a wrong answer, the fault could lie with the poorly performing UQ method or with the model itself having incorrect knowledge. We suggest a way to disentangle the two: compare several UQ methods on the same model and data. When one method performs well and another fails, the difference points to the method rather than the model. To compare the UQ methods we will use the two metrics defined earlier: ASCE and ROC-AUC.

To make the comparison more informative, we run each method on distinct knowledge domains: history, medicine, law, mathematics, etc., rather than on a single mixed test set. Detommaso et al. (2024) first noted that the TF softmax calibration varies by domain, most likely due to differences in knowledge about different topics, which they called multicalibration. They highlighted that regular calibration (Definition 4) requires the scores to be calibrated over the whole dataset marginally, e.g., a 0.9 score on average should correspond to 90% correctness on average, while in reality the scores can still be very off on some parts of the data - there might be clear subsets with significantly higher or lower calibration error.

We expand their insights by examining all the UQ methods, not only the TF softmax score, through this lens of multicalibration. Furthermore, we note that calibration and effectiveness may vary not only across domains but also across different methods that we use. On some domains, method A may be better than method B, while the opposite may be true on other domains. This is because these methods work in different ways and operate under different assumptions.

This experiment may help us answer our research questions. If the domain shifts are significant, this is an argument that UQ methods cannot be trusted on new, untested data. By investigating in which domains the confidence scores are reliable and in which ones they aren't, we may find general rules about when to trust them. If any method significantly outperforms others on all domains, we can further hypothesize about which mechanism elicits true confidence scores from the model.

5.2 Datasets and Models

5.2.1 Datasets

Our experiments use closed-book question answering in a free-form format: the model must produce its own sentence, not select from multiple choice. We chose this setup as it's the most similar to popular LLM chat systems like ChatGPT (Bahak et al., 2023) and poses a tougher test for uncertainty quantification. In particular, we focus on two popular benchmarks:

MMLU (Hendrycks et al., 2021). Originally multiple-choice, but following Farquhar et al. (2024), we convert it to a free-form answer format by asking a model for a full sentence as a response. See Appendix B for more details.

TriviaQA (Joshi et al., 2017). Originally a retrieval-augmented QA benchmark where each question has a corresponding source document useful for answering, we strip this extra context and ask the model the question without access to it.

To enable per-domain analysis, each dataset was categorized into six broad categories, such as *ethics* and *history*, further described in Appendix B. Next, the full datasets were labeled using the OpenAI GPT-4o-mini LLM (OpenAI, 2024), with non-matching samples being discarded. Finally, we selected balanced samples of both TriviaQA and MMLU datasets: 230 samples per category for TriviaQA and 229 per category for MMLU.

This yields a total of 2754 evaluation questions: 1380 from TriviaQA and 1374 from MMLU. An example of a true TriviaQA question, as well as model responses, can be found in Figure 5.7a. All items are reserved for testing. We perform no additional fine-tuning or in-domain calibration, which we acknowledge as a potential limitation.

5.2.2 Models

In our experiments, we use two instruction-tuned (post-alignment) open-source LLMs: (1) Llama-3.1-8B-Instruct (Dubey et al., 2024) and (2) Gemma-2-9B-IT (Riviere et al., 2024). These models will be presented with questions to which they will generate answers. For each answer, we will obtain confidence scores with selected UQ methods.

Both models are run on a single NVIDIA A100 GPU. Environmental metrics for these runs are reported in chapter 9. We chose these open-source models due to limited computational power, better reproducibility and access to token probabilities, which are necessary for computing the TF Softmax score. Additionally, we believe that using small to moderate-sized models may provide more insight into UQ methods. With larger and more powerful models, it becomes more difficult to find cases where they are wrong, resulting in less signal for analysis. To generate text using these language models, we used the temperature sampling method, described in detail in Appendix A, with a low temperature of 0.1 for all runs except Semantic Entropy, where we used a value of 1.0 recommended by Farquhar et al. (2024).

Answer correctness was determined by querying GPT-4o-mini via the OpenAI API (OpenAI, 2024) that was provided with the question, generated answer and ground truth from the dataset. For grouping semantically equivalent answers required for Semantic Entropy, we used the same GPT-4o-mini model, for each pair of candidate answers, asking whether they mean the same given the question. Specific prompts and parameters used are provided in Appendix C.

These datasets and models give us a balanced mix of questions while keeping the compute budget practical, allowing us to study how each uncertainty quantification method behaves across diverse domains.

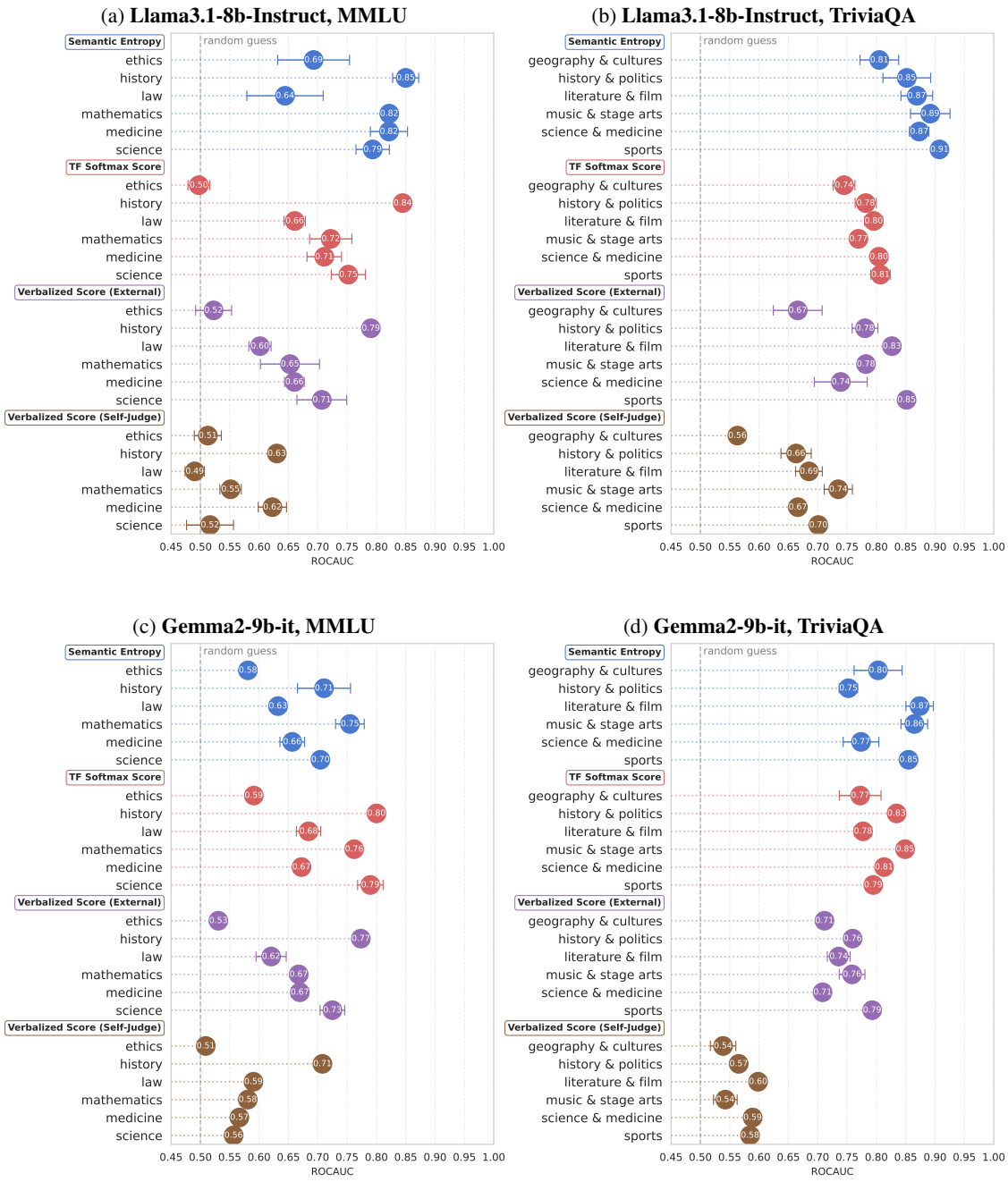


Figure 5.1: Comparison of UQ Methods effectiveness on MMLU and TriviaQA datasets across UQ methods and domains. Measured as ROCAUC in the binary classification of false generations. Each experiment was repeated 3 times - we provide the mean values and 95% confidence interval whiskers.

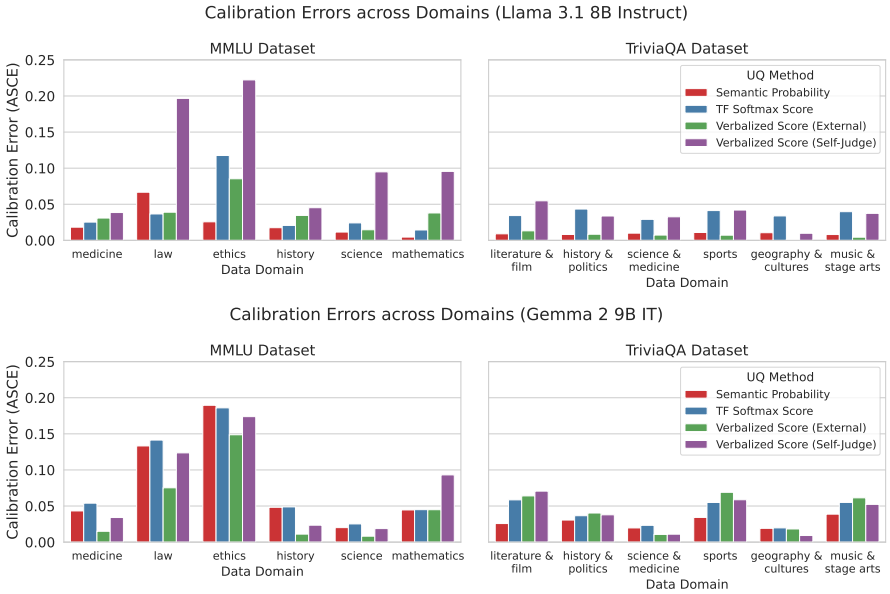


Figure 5.2: Mean ASCE calibration errors across models, datasets, and domains.

5.3 Domain and method-wise inconsistencies

The results in Figure 5.1 and Figure 5.2 confirm that domain shifts are significant. TF Softmax, for example, ranges from random (ROC-AUC 0.50 on *ethics*) to very effective (0.84 on *history*) on the same Llama-3 model, a difference of 0.34 points.

Across both models and datasets, the Verbalized scores are usually the least informative. The external variant reaches only 0.52 ROC-AUC on *ethics* (Llama) and 0.52 on *stage art* (Gemma), yet climbs to 0.85 on *sports*, showing that VC can be informative on some domains. Self-judge variant is worse than the External one - on Llama, its confidence values are on average 0.22 percentage points higher than the VC External for the same example, showing systematic overconfidence in self-assessment and strong sensitivity to how the task is framed.

Semantic Entropy is the most reliable overall, but not completely immune to the domain effects. On *sports* questions in TriviaQA, it achieves great ASCE 0.01 and ROC-AUC 0.91 (Llama), yet on *law*, its calibration error rises to 0.06, and TF Softmax beats it in effectiveness. In the *history* slice of MMLU and TriviaQA, SE is less effective than external VC for Gemma.

Taken together, our results show that no single method captures model confidence equally well across domains. This result exposes a risk in using a single UQ metric with a general-purpose LLM: even after extensive benchmark testing, there is no guarantee that the selected confidence score will remain trustworthy on unseen topics. Misleading scores (especially over-confident ones) may be worse than no score at all because they give users a false sense of certainty. We return to the ethical implications of this point in chapter 9.

Why are the domain differences so large? Detommaso et al. (2024) mentions question/domain difficulty as a potential cause. Domains such as *ethics*, where all methods perform poorly, may contain more ambiguous or difficult questions where the model may provide incorrect (different from the ground truth) answers confidently. Difficulty could also explain differences between different UQ methods: maybe they thrive at different difficulty levels: some may excel on easy questions, others on hard ones. To test these ideas, the next section introduces a third variable, in-domain accuracy, and asks how effectiveness and calibration change with task difficulty.

5.4 Domain Difficulty vs UQ Score Quality

We plot ROCAUC and ASCE against domain accuracy in Figure 5.3 through Figure 5.6. For most methods (all except TF Softmax on Llama), ASCE correlates strongly with domain accuracy: the easier the domain, the lower the error. VC Self-Judge is an extreme case (Spearman $\rho = -0.97$ for Llama, -0.98 for Gemma). The correlation, however, says more about domain difficulty than about method quality. Because VC Self-Judge always emits very high numbers (we verified that the average confidence score is always higher than 0.89 for all domains), it looks well-calibrated only when the domain itself has many correct answers.

Method effectiveness can give us more insight. The two hardest domains, *law* and *ethics*, yield the lowest ROC-AUC for every score. We find a moderate, positive correlation between answer accuracy and ROC-AUC ($0.30 \leq \rho \leq 0.60$ for most scores) for nearly all domains and methods. We see that every UQ method becomes substantially more informative once domain accuracy passes roughly 0.70. If the domain is difficult or ambiguous, the model may simply "guess" the answer, then the obtained confidence scores cannot be used to discriminate between correct and incorrect guesses.

While difficulty can be used to explain the domain shifts, it does not explain why Semantic Entropy so often outperforms Verbalized Confidence inside the same domain. To isolate that effect, the next section takes a closer, qualitative look at these methods, comparing the predictions they give and the failure modes.

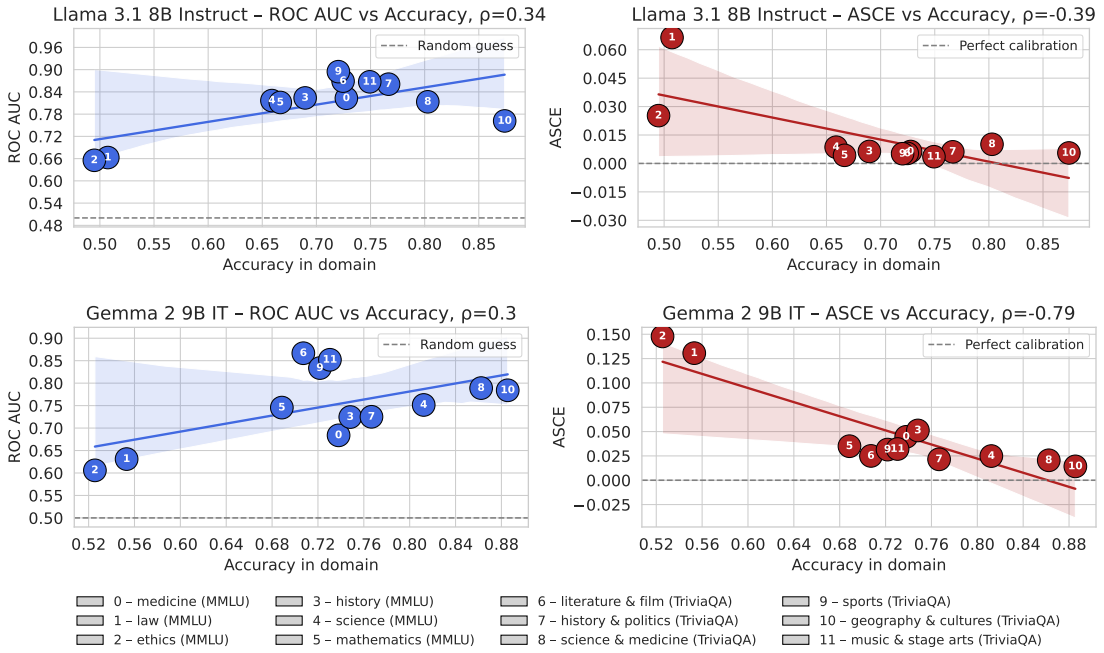


Figure 5.3: Semantic Entropy for ROCAUC, Semantic Probability for ASCE. Spearman correlation ρ in the title.

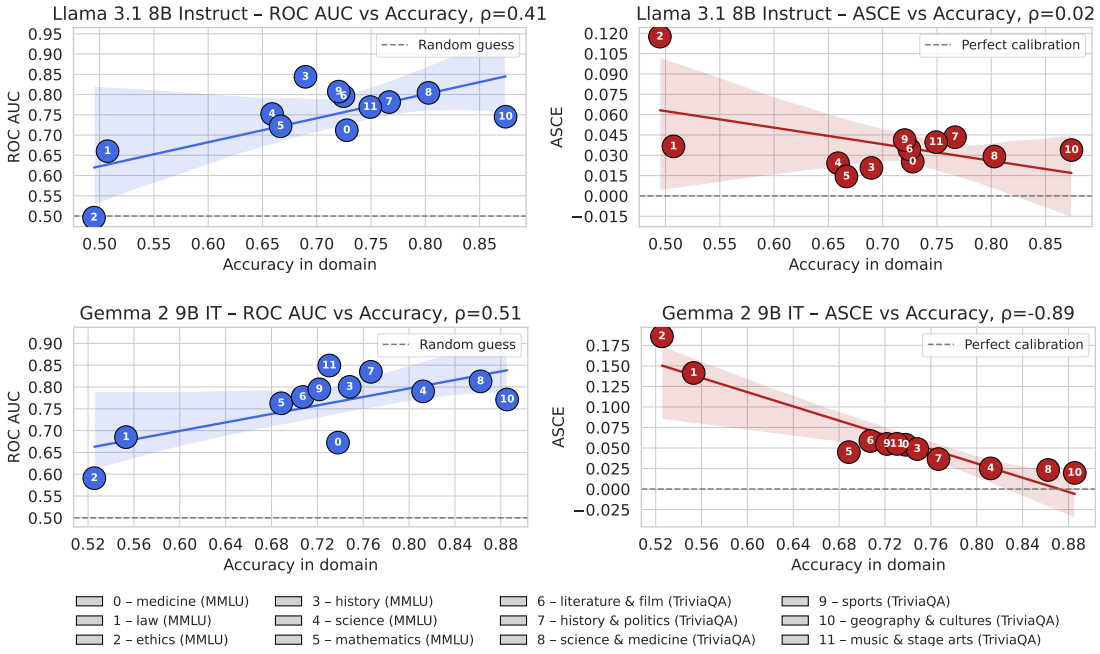


Figure 5.4: TF Softmax Score, Spearman correlation ρ in the title.

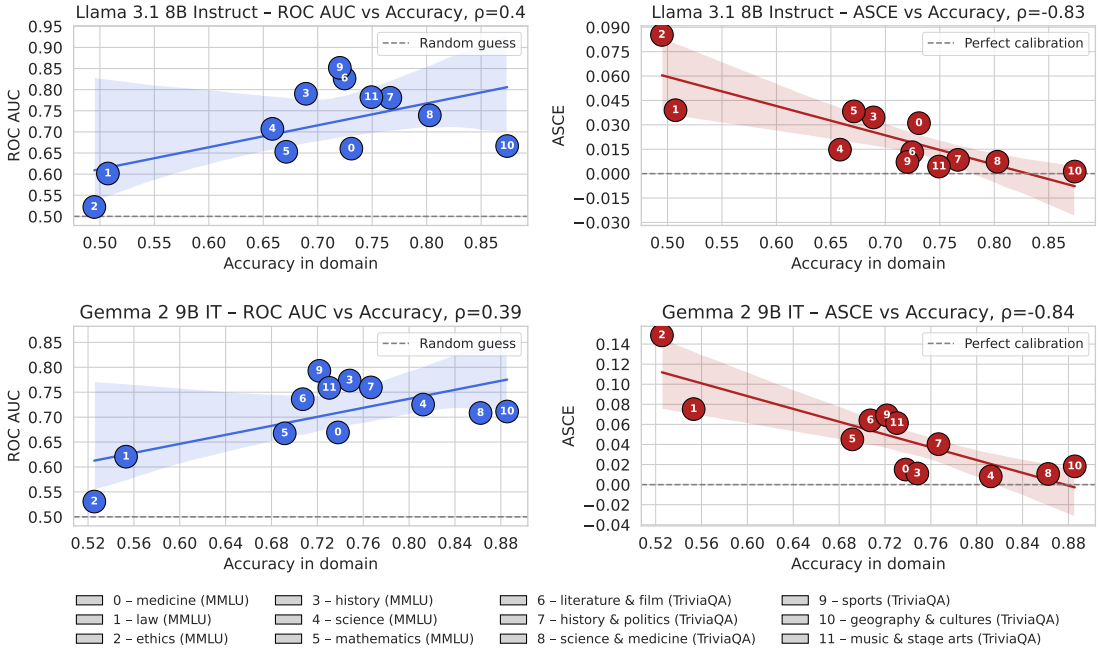


Figure 5.5: Verbalized External Score, Spearman correlation ρ in the title.

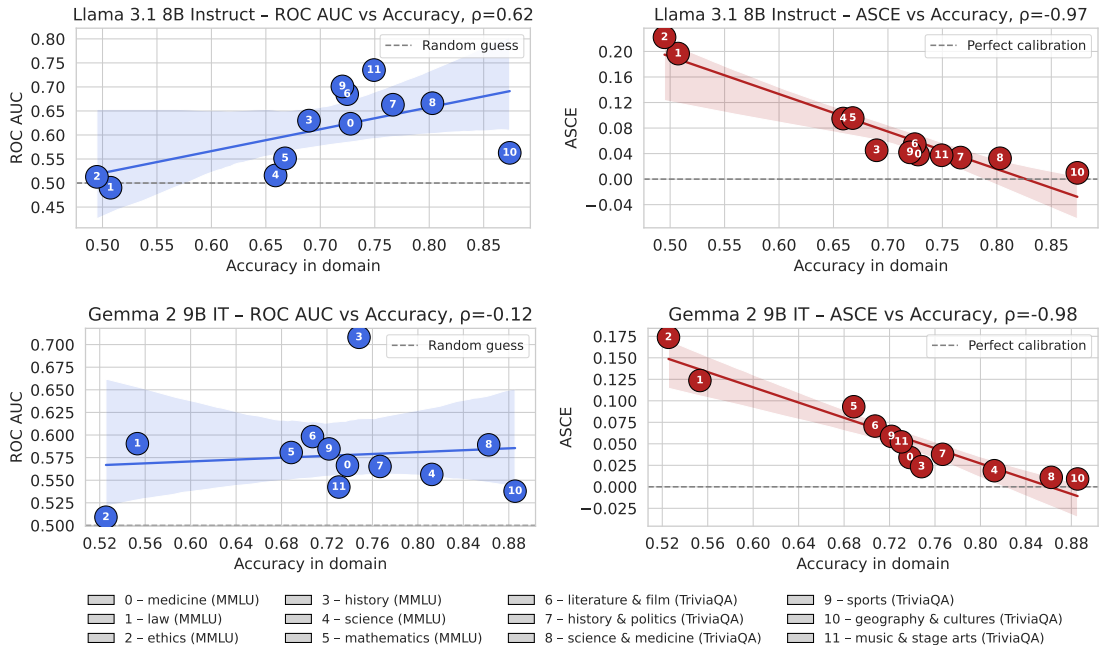


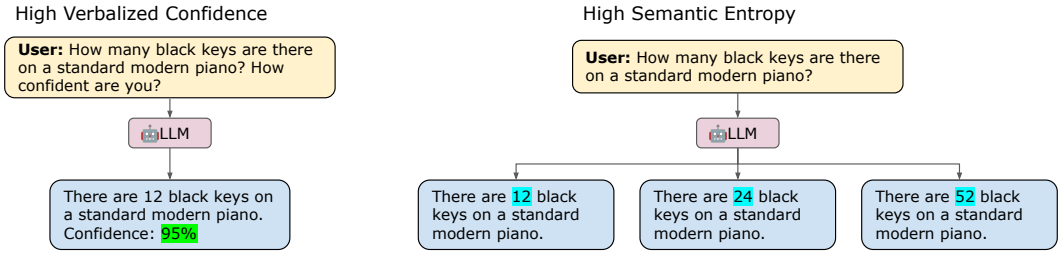
Figure 5.6: Verbalized Self-Judge Score, Spearman correlation ρ in the title.

5.5 Verbalized Confidence vs Semantic Entropy: Finding Failure Modes

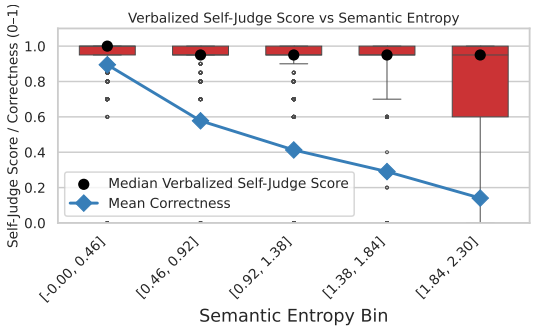
The previous section showed that inconsistencies between domains and methods are large and that no single score is safe to deploy without validation. In-domain accuracy explains part of the domain effect, yet it does not account for the wide gap we still observe between Semantic Entropy (SE) and Verbalized Confidence (VC).

Figure 5.7a presents a concrete example. Semantic Entropy is high: the model produces a different number each time, while VC Self-Judge reports about 95 percent certainty. Which number should we trust? We argue for SE on two grounds. First, SE achieves higher ROC-AUC across most domains, so it discriminates better between correct and incorrect answers. Second, the sampled responses are drawn in proportion to the model's own token probabilities, giving SE a direct link to the internal uncertainty (section 3.3)

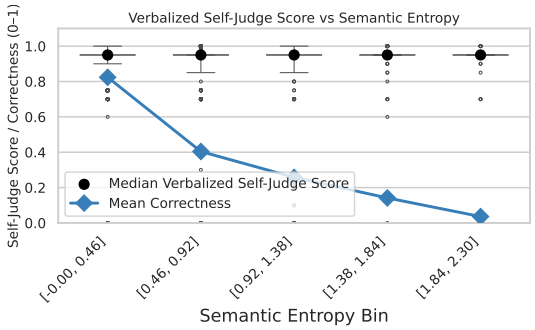
This pattern generalizes in Figure 5.7. As Semantic Entropy rises (moving right on the x-axis), the fraction of correct answers in each bin goes down, to about 20 percent in the last bin, yet the median VC Self-Judge score stays in the ninety-plus range. VC, therefore, fails to reflect the uncertainty properly, while SE succeeds by approximating the entropy of the answer distribution $P'(a \mid q)$ defined in section 3.3.



(a) Real example where High verbalized confidence conflicts with high semantic entropy (TriviaQA, Llama 3.1 8B)



(b) LLama 3.1 8B Instruct



(c) Gemma 2 9B it

Figure 5.7: Mean Answer accuracy and Median Verbalized Confidence (Self-Judge) across growing Semantic Entropy bins. Black circles represent median values, red boxes determine the 0.25 to 0.75 quantile. For Gemma, the majority of the VC Scores are equal 0.95, even when the average accuracy is low.

The evidence so far suggests that SE is better suited to surface the model's uncertainty, while Verbalized Confidence is less informative and overconfident. We, however, find this comparison not entirely fair: Semantic Entropy utilizes the generation of extra tokens at test-time to estimate the predictive distribution and surface possible alternatives, while Verbalized Confidence is restricted only to the one generated answer path. We hypothesize that equipping Verbalized Confidence with a predictive space exploration mechanism similar to SE's could lead to significant improvements in the confidence score quality.

To put that into a test, in the next chapter, we will perform a set of experiments where LLM does test-time chain-of-thought reasoning before providing a final answer and verbalized confidence estimates. This set of experiments will help us better understand if the model has access to internal (latent) uncertainty signals, which is necessary for VC, or if exploration is needed. We will also conduct a fair comparison, testing how VC compares with SE when test-time budgets are matched.

Chapter 6

Test-time Token Sampling as a Source of Confidence Signal

The cross-domain study in Chapter 5 left a puzzle. Semantic Entropy was more reliable than Verbalized Confidence, even when both ran on the same model and the same questions. We hypothesize that SE's higher quality comes from a crucial detail: it samples from the model many times at test time, exploring other alternative generation paths: this is illustrated in the Figure 3.3.

That observation raises a follow-up question. Is this extra reliability a unique property of SE, or is it simply what happens when we generate more continuations from the model? In other words, could a different form of test-time exploration make Verbalized Confidence just as good?

This chapter's goal is to answer these questions. Jurayj et al. (2025) has shown, that chain-of-thought reasoning Wei et al. (2023) can be used as an effective way of test-time sampling. They show that forcing the model to reason before answering improves the calibration of the final answer's joint token-level probability. Furthermore, chain-of-thought reasoning can improve answer accuracy (Wei et al., 2023), which we identified as a potential factor in UQ effectiveness in the previous chapter.

That's why in this chapter we employ Verbalized Confidence with chain-of-thought reasoning, as an alternative way of sampling from model's predictive distribution. We enforce the model to generate a chain of thought before stating both the final answer and its confidence, and compare the quality gains to Semantic Entropy.

This setup also lets us probe a deeper issue: by analyzing how the Verbalized Confidence changes due to reasoning, and thanks to later ablations of the reasoning trace, we can try to infer if the model holds a latent sense of uncertainty, or are its verbalized scores merely a summary of whatever appears in the reasoning trace. The experiments that follow are designed to answer both points.

6.1 Experiment Design

Our goal in this chapter is to find out whether chain-of-thought reasoning can serve as the same kind of test-time exploration that makes Semantic Entropy effective, and by doing so lift Verbalized Confidence to a comparable level. We pose two competing pictures as our work hypotheses:

Latent-belief view The model has an internal belief of being correct, which is not expressed in the generated text itself. Even if this internal belief is not perfect, this is where the model takes the Verbalized Confidence from, and adding more reasoning steps doesn't change the score significantly.

Self-sampling view The model has no direct access to such a belief. It forms its confidence only after seeing what it has generated. Reliable Verbalized Confidence appears only when the model explicitly surfaces (generates) samples from its own predictive space, in a manner similar to Semantic Entropy.

The most recent studies report that reasoning-tuned models yield better-calibrated VC scores than their non-reasoning counterparts (Hammoud et al., 2025; Zeng et al., 2025), but none have compared VC + reasoning with SE, assessed how the reliability changes with controlled length of reasoning budgets, or ablated the reasoning trace to see the VC change.

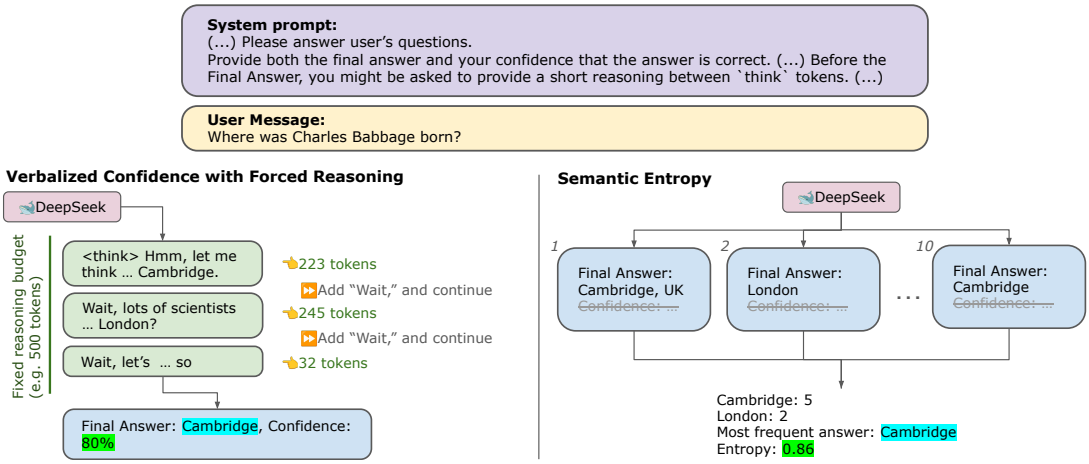


Figure 6.1: **Two tested methods of obtaining Final Answer and Confidence** - Verbalized Confidence with Forced Reasoning (in this chapter referred to as VC) works by prompting the model to reason for longer-until the fixed budget is exhausted - before stating the answer and confidence. Semantic Entropy (SE) obtains 10 independent answers that are later clustered semantically to identify the most frequent one, and to calculate the entropy in the answer distribution.

We address that gap with a set of experiments that (i) force the model to reason before stating its confidence, (ii) compare the resulting effectiveness and calibration with Semantic Entropy under matched compute budgets, and (iii) analyze and ablate the reasoning traces to see where the VC uncertainty signal resides. The detailed setup follows in the next sections.

6.2 Metrics and Methods

We compare Verbalized Confidence (Self-Judge) employed with chain-of-thought reasoning, further referred to just as Verbalized Confidence (or VC), with Semantic Entropy without reasoning. Using reasoning with Semantic Entropy could be an interesting experiment, but due to even larger computational requirements, we leave it for future research. For clarity, we decided to report only the effectiveness (ROC AUC) in this chapter, as in the previous chapters, it proved to disentangle the methods' informativeness from the answer accuracy better than the calibration error.

6.3 Datasets and Models

Data Sources. Because long-trace experiments are computationally expensive, we built a small (270 samples) but diverse benchmark instead of using full datasets. We sampled questions from five popular, open-source datasets: TriviaQA, MMLU, and SimpleQA for fact retrieval (Joshi et al., 2017; Hendrycks et al., 2021; Wei et al., 2024), plus GSM8K and AIME-2024 for mathematical reasoning (Cobbe et al., 2021; AIME, 2024). Our goal is open-ended QA in natural language, so we stripped away multiple-choice options in MMLU, and any figure references in AIME-2024, manually discarding questions that could not stand alone after this edit, such as *"Which of the following is true?"*. Every surviving example was then hand-labeled with its knowledge domain and the skills needed to answer it, such as "Fact Retrieval" or "Mathematical Reasoning". Full sampling details and the final label distribution appear in Appendix B.

Model and Prompts. We chose Deepseek-R1-32B[1] (DeepSeek-AI et al., 2025) following a similar experimental setup of Jurayj et al. (2025) for its strong reasoning capabilities at a manageable model size. Furthermore, it is one of the most popular open-sourced reasoning-tuned models. All experiments described in this chapter were run on two NVIDIA A100 GPUs.

The prompts we used in this experiment differ from the previous one, as we designed them to encourage the LLM to do reasoning, as authors of DeepSeek R1 recommend (DeepSeek-AI et al., 2025). We provide an illustration of the prompting and inference we adopted in Figure 6.1. Across all setups, we used a single system prompt that directs the model to (1) think step by step, and then (2) provide a final answer along with a Verbalized Confidence score. The full prompt text, as well as a real interaction example, is available in Appendix C.

In our experiments, we want to control the length of the reasoning chain (computational budget for one question) to get more insight into how the number of tokens generated impacts the metrics. To regulate it, we applied the budget-based truncation method of Muennighoff et al. (2025) also employed by Jurayj et al. (2025): we choose a fixed budget, and: 1) When the reasoning budget is exhausted (or set to zero), the chain terminates immediately, 2) If the budget remains, the system appends `Wait,` tokens, and asks the model to generate more tokens. As in the previous chapter, we used temperature sampling (described in Appendix A with temperature equal to 0.1 for Verbalized Confidence (and reasoning), and 1.0 for parallel sampling in Semantic Entropy experiments.

6.4 Impact of Reasoning on Verbalized Confidence

Figure 6.2 shows final-answer accuracy, UQ effectiveness, and average stated confidence for correct and incorrect answers as a function of the reasoning budget, with Semantic Entropy shown for comparison. We make the following observations:

O1: Granting just 100–500 reasoning tokens raises accuracy $41\% \rightarrow 63\%$ and boosts verbalized-confidence ROC-AUC $0.56 \rightarrow 0.80$. The mean confidence for correct answers remains high ($\approx 95\%$), whereas confidence for errors falls down to $\approx 80\%$.

O2: For fact-retrieval questions (Fig. 6.2b), accuracy does not improve with longer reasoning budgets, yet calibration continues to improve with additional tokens. We can reach very long reasoning traces for fact-retrieval questions thanks to the employed forced reasoning technique (Muennighoff et al., 2025) presented in Figure 6.1.

O3: Verbalized Confidence is initially weaker than superior Semantic Entropy but reaches near parity at 200 tokens for fact retrieval and 3,500 tokens for mathematical items, while maintaining higher answer accuracy due to the reasoning process.

These trends confirm that allocating test-time compute to reasoning is essential for reliable uncertainty estimates, and extended CoT effectively mitigates DeepSeek's over-confidence without sacrificing performance. The sheer scale of the improvement in effectiveness: from near-random 0.56 ROCAUC to 0.88 suggests that there is no latent uncertainty information available for the model, and self-sampling is necessary to obtain a good uncertainty estimate.

[1] `deepseek-ai/DeepSeek-R1-Distill-Qwen-32B`

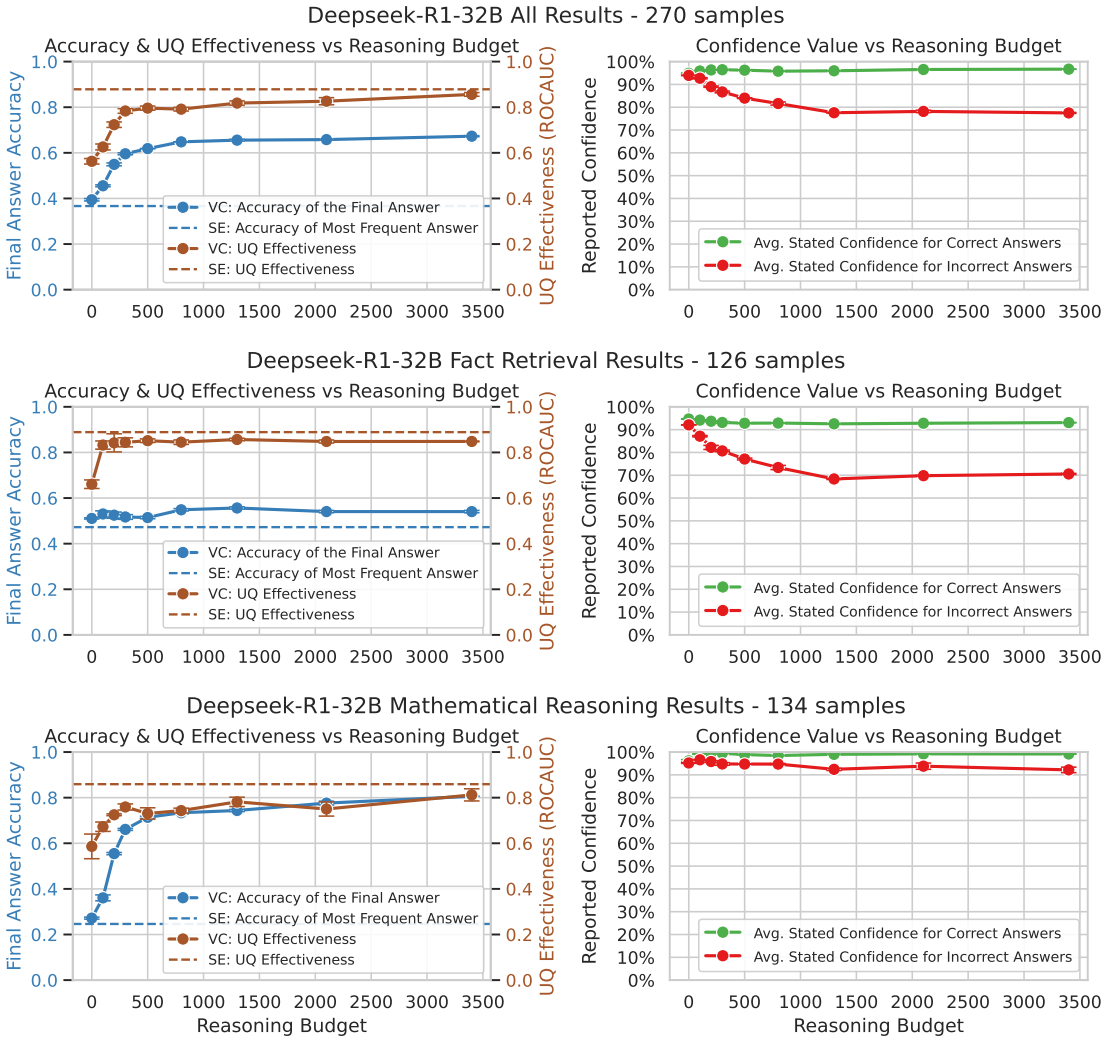


Figure 6.2: **Effectiveness and Accuracy of Verbalized Confidence with Forced Reasoning vs Semantic Entropy.** (a) Full overview. (b) Fact retrieval results. (c) Mathematical reasoning results. Note: The remaining 10 samples not falling into the Fact Retrieval or Mathematical Reasoning categories are included in the Full overview but not presented as separate plots.

	Avg. Tok	Acc.	ROC-AUC
All 270 Samples			
No reasoning VC	15.4±0.1	0.39±0.01	0.56±0.01
200-tokens VC	222.7±0.6	**0.55**±0.01	0.72±0.01
SE	218.2±0.4	0.37±0.04	**0.88**±0.04
Fact Retrieval			
No reasoning VC	15.8±0.2	**0.51**±0.00	0.66±0.02
200-tokens VC	222.7±0.1	**0.53**±0.01	0.84±0.04
SE	177.2±19.4	0.47±0.00	**0.89**±0.03
Math Reasoning			
No reasoning VC	14.8±0.1	0.27±0.00	0.59±0.05
200-tokens VC	221.5±1.5	**0.56**±0.01	0.73±0.00
SE	241.4±15.3	0.25±0.07	**0.86**±0.03

Table 6.1: **Budget-fair comparison of Semantic Entropy (SE) and Verbalized Confidence (VC).** Average generated tokens (*Avg. Tok*), answer accuracy (*Acc.*), and calibration quality (*ROC-AUC*). SE has a much larger budget than zero-shot VC, explaining its superior calibration. When budgets are matched (≈ 220 tokens per sample), Verbalized Confidence is close to SE's ROC-AUC on fact-retrieval and yields higher accuracy across both domains.

6.5 Budget-fair comparison between Verbalized Confidence and Semantic Entropy

In the previous section, we showed that there may be no latent confidence accessible to the model, and the quality of the confidence estimate could be attributed to the number of samples drawn from the predictive distribution. If that is true, we expect to see a similar performance of Semantic Entropy and Verbalized Confidence when the two methods have the same budget for sampling from the model. We computed the average number of tokens per sample used by Semantic Entropy and compared it with the VC that uses the closest token budget. We added VC with no reasoning for comparison - all results can be found in Table 6.1

We see that Semantic Entropy outperforms zero-shot Verbalized Confidence, but it spends roughly fourteen times more test-time compute tokens. With a comparable 200-token budget, Verbalized Confidence: (1) almost matches SE's calibration on fact-retrieval tasks (ROC-AUC 0.84 ± 0.04 vs. 0.89 ± 0.03), and (2) delivers substantially higher answer accuracy on both fact-retrieval (+0.06) and mathematical reasoning (+0.31). On mathematics, SE remains better calibrated, yet the accuracy gain coming from longer reasoning trace may be preferable in many applications. We provide additional results across 5 source datasets in Figure C.5.

Overall, with test-time compute matched, Semantic Entropy still leads in calibration, suggesting it squeezes a bit more signal from each token. Yet the margin is now far smaller than in the zero-reasoning baseline. This further strengthens the claim that, while UQ heuristics differ in efficiency, the act of sampling itself is the critical ingredient for reliable uncertainty.

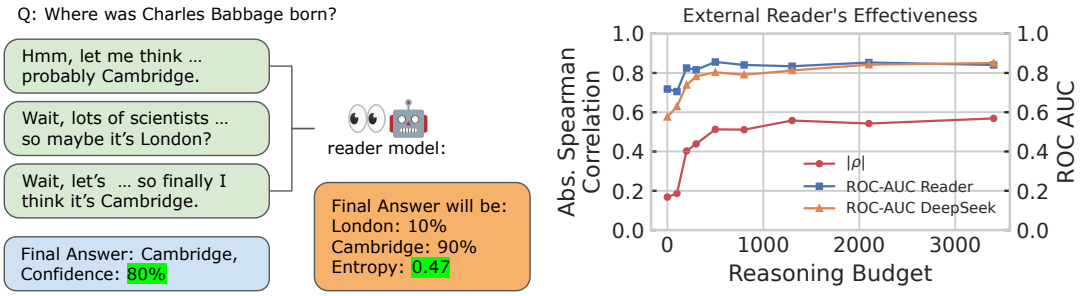


Figure 6.3: **Separate reader matches the reliability of DeepSeek's own Verbalized Confidence by just looking at the reasoning trace.** With more reasoning tokens, the agreement between them (measured as absolute Spearman correlation) increases, and the effectiveness of both scores changes similarly.

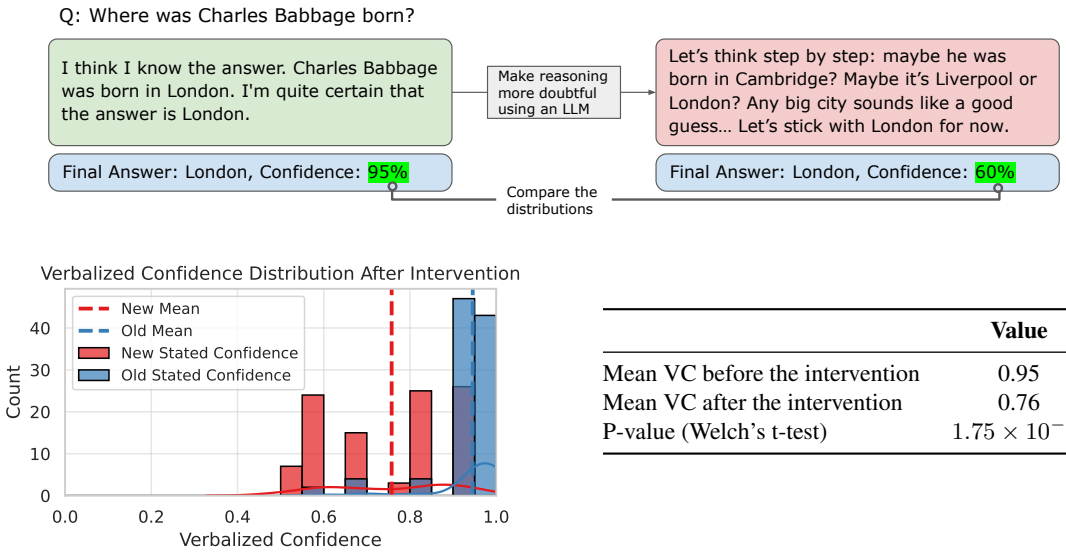
6.6 Reader-Model Recovers Uncertainty Signals

If there is no hidden latent variable from which Verbalized Confidence is drawn, then the reasoning trace has to contain all the uncertainty information needed to explain DeepSeek's final score. We can verify it using an external "reader" model that tries to predict how uncertain DeepSeek is by reading its reasoning trace. If the reader can predict DeepSeek's confidence just based on what's written in the trace, that's a strong signal that there is no additional hidden latent confidence.

Figure 6.3 illustrates our experimental setup and results. As a reader model, we used OpenAI's GPT-4o-mini (OpenAI, 2024), we provide more information about the prompt and method used to obtain the probabilities in Appendix C. We display (i) the absolute Spearman correlation $|\rho|$ between DeepSeek's self-reported confidence and the reader entropy H_{reader}, and (ii) the effectiveness of both scores.

With no reasoning tokens exposed, the correlation between Reader's and DeepSeek's scores is low, however, with more reasoning tokens, the effectiveness of the reader goes up in tandem with DeepSeek's effectiveness, and the correlation between the two goes up. At 3.4 k tokens, DeepSeek reaches ROC-AUC = 0.851 and the reader 0.841 with $|\rho| = 0.57$, indicating that almost the entire confidence signal is now accessible in the trace.

These results show that all the uncertainty signals used by the Verbalized Confidence surfaced (was generated) during the reasoning process, and there is no hidden internal confidence belief that is not exposed in the generated reasoning trace. This further strengthens our belief that model's Verbalized Confidence is just a "description", or a "reader model"of what was surfaced previously, and without any reasoning, there is nothing to read or describe, explaining poor effectiveness in the previous chapter. We conducted one more experiment that confirms this is the case in our final ablation experiment, where we alter the content of the reasoning trace and see how it affects the final Verbalized Confidence.



	Value
Mean VC before the intervention	0.95
Mean VC after the intervention	0.76
P-value (Welch's t-test)	1.75×10^{-23}

Figure 6.4: **Altering the reasoning chain makes the model more doubtful, despite answering correctly.** This experiment was conducted on 100 random samples that were answered correctly by the DeepSeek, with a reasoning chain length of 1300 tokens.

6.7 Altering Reasoning Trace Changes Confidence

If this is true, that the model is just reading its own generation, and selecting what Verbalized Confidence score would match the level of expressed uncertainty and number of alternatives, then altering the content of the generation should also greatly affect the final score. We selected 100 questions that DeepSeek answered correctly after 1300 reasoning tokens and asked GPT-4o-mini OpenAI (2024) to rewrite each trace in a more doubtful tone:

> (...) You will be provided a question and a real reasoning trace, Create a new, similar reasoning trace of a much doubtful persona that thinks out loud about many possible answers.

The full prompt can be found in Appendix C.

We then let DeepSeek continue from the altered trace to produce its final answer and confidence. Figure 6.4 shows the result: the mean Verbalised Confidence falls from 0.95 to 0.76 ($p = 1.75 \times 10^{-23}$, Welch's t-test), even though the model had correct knowledge to answer the question originally.

While this experiment is limited, as the process of altering the reasoning chain may actually change the final answer for some questions, the observed effect is very large. Together with the reader study, this makes us believe that the Verbalized Score depends on what is written in the chain of thought, not on an inaccessible internal belief. Reliable confidence, therefore, requires test-time exploration that surfaces alternative possibilities, whether by parallel sampling (Semantic Entropy) or by extended reasoning that the model itself can read and summarize.

Chapter 7

Discussion

7.1 Recap of the Previous Chapters

Large language models can hallucinate while seeming highly confident. Our running example: “The maximum daily ibuprofen dose is 3000 mg” is expressed in a confident tone, but can we trust this answer? Existing Uncertainty Quantification (UQ) methods provide us with confidence scores: now the user realizes that the model is ”70% sure”. But there is a new risk: user may trust this number, yet they have no way to know whether that number is grounded in anything real and reliable - maybe the model completely guessed the ”3000 mg” and attached a completely non-related label ”70%”.

This thesis asked a simple question: Assuming we use a UQ method to obtain a confidence score, what does the number actually measure, and can we trust it?

Chapters 3-4 surveyed three families of UQ methods: True/False Softmax, Semantic Entropy (SE), and Verbalized Confidence (VC), as well as two evaluation axes, ROC-AUC and ASCE. Chapter 5 presented how these methods perform on different data domains (such as medicine, sport, ethics), and what are the common trends and failure modes, while Chapter 6 explored the idea of test-time token exploration as a source of reliable confidence scores.

7.2 Key Findings

1. Domain can matter more than method. Differences in the effectiveness across domains for one UQ method can be even higher than between different methods (as high as 0.34 ROC-AUC). Semantic Entropy was the most reliable overall, yet it dropped to second or third place on domains like *law* and *history*.

2. Difficulty sets a lower bound for UQ effectiveness. When the domain is difficult or ambiguous, and in-domain answer accuracy falls below a certain threshold, all scores (including SE) lose discriminatory power. In other words, if the model is simply guessing for all the questions in the domain, we can’t use UQ methods to differentiate between correct and incorrect guesses.

3. Verbalized Confidence is unreliable at low test-time compute. VC Self-Judge was usually over-confident and sometimes barely better than random. The score improved only when the model was forced to reason for hundreds of tokens before answering and giving confidence estimate.

4. Reliable scores require exploration of the predictive distribution. SE achieves this with parallel sampling, which is why it’s so effective. VC can do the same after a sufficiently long chain of thought. Both approaches collapse when there is not enough test-time sampling.

5. Verbalized Confidence is based more on what’s generated so far than internal confidence belief. An external reader model, just by looking at DeepSeek’s reasoning trace, can express DeepSeek’s confidence with the same reliability (within 0.01pp ROC-AUC difference). Altering that trace caused the DeepSeek to revise its confidence, showing there is no hidden confidence belief state accessible to the model, beyond what was sampled in the reasoning chain.

7.3 Practical Guidelines

Reliable confidence for today's medium-sized LLMs appears to need two ingredients:

1. **Knowledge.** The model must know enough about the domain to be able to differentiate between guesses and knowledgeable answers. Larger models and training datasets, domain-specific fine-tuning, and reasoning can help in this matter.

2. **Exploration.** The model must sample its predictive space: either explicitly (SE) or implicitly (long reasoning), before it can judge how likely its answer is to be true.

We argue that future UQ research should focus less on inventing new black-box heuristics and more on finding ways to efficiently sample and explore all internal token probabilities. Currently, it seems like an extra few hundred tokens generated at test time are necessary for a decently performing UQ method. We thus enumerate the following practical guidelines:

- **Per-domain Evaluation.** Evaluate confidence scores quality across many domains. A global metric averages can hide large errors in small but critical slices, and the scores may perform surprisingly bad on new, difficult domains.

- **Budget matters.** Use Semantic Entropy or generate a 200-token minimum reasoning chain before trusting any confidence score.

- **Fallback to retrieval when possible.** Retrieval-augmented generation reduces hallucination risk by relying on external verified knowledge. While this thesis describes a setting where such a method can't be used (e.g., on-device LLMs, no internet access, out-of-domain questions with no relevant documents), existing research shows that retrieval is key in reducing hallucinations, which is further discussed in Appendix A.

7.4 Final Remarks

Even the best UQ method cannot be trusted outside the domains on which it was benchmarked. Until models acquire broader factual coverage, practical systems should restrict confidence scores to narrow, tested domains and use only methods that do test-time exploration. Our work exposes a paradox for simple and user-friendly methods like Verbalized Confidence - this method's appeal relies on the fact that it is low-compute and easy to use on new data, even for a non-technical person. At the same time, we see that in such settings it simply doesn't work - it has to be restricted to just a few domains and more compute is needed, sacrificing its appeal. We hope the tools and findings in this thesis move the field one step closer to LLMs that are not only fluent but also responsibly self-aware. Ethical and environmental implications of what we described are discussed in chapter 9.

Chapter 8

Limitations and Future Work

Model coverage We evaluated two small-to-medium instruction-tuned models and used a single 32B reasoning model in the reasoning study. Larger models might possess broader knowledge or different alignment behaviour, and our conclusions, particularly about the amount of test-time compute needed, should be confirmed at that scale. We currently believe that any autoregressive model faces the same bottleneck: until it generates enough tokens to explore its predictive space, it cannot surface a trustworthy self-assessment, but this remains to be tested on other models.

Dataset scope Our analysis in Chapter 5 relied on MMLU and TriviaQA, expanded into six LLM-annotated domains each, while conclusions in Chapter 6 come from one, relatively small dataset divided into 2 sub-categories. All our conclusions come from this relatively small data suite and may not generalize to other types of questions or benchmarks. Furthermore, parts of the test-time compute analysis in Chapter 6 rely on the aggregated metrics, without considering domain-wise performance, which is a big limitation.

Future work should replicate the experiments on additional QA corpora and verify if observed trends occur on different domains.

Sources of uncertainty Our analysis focused on the uncertainty that arises from token sampling and alternative answers. Other sources - such as uncertainty based on the retrieved documents' content, or uncertainty from multiple models ensemble - were not tested and not hypothesized in this work, possibly resulting in a false sense that some methods are useless without test time exploration.

Method breadth The test-time compute comparison in Chapter 6 covered only two methods, VC and SE. Other methods, such as TF Softmax Score, Perplexity, or different ways of prompting the LLM in VC could be included in the future benchmarks. Maybe some methods could prove more efficient in the budget-effectiveness comparison.

Exploration strategies Even when budgets were matched, Semantic Entropy remained a bit better than reasoning-enhanced VC. This shows that while predictive space exploration (sampling or reasoning) is key, how we explore it matters too. We hypothesize that there might be better and more efficient ways of exploring the space, such as tuning the sampling parameters in Semantic Entropy (such as temperature), or adapting existing uncertainty-aware decoding techniques.

Addressing these limitations and expanding this work with suggested experiments would increase our understanding of where confidence estimates originate and how best to surface them in real systems.

Chapter 9

Ethical and Environmental Considerations

9.1 Increased Environmental Strain

Jegham et al. (2025) estimate that global use of OpenAI's GPT-4o could lead to the evaporation of 1,334,991 kilolitres of fresh water for cooling and power generation (enough to meet the annual drinking needs of about 1.2 million people), and emit more than 130,000 tonnes of CO_2. This thesis adds, in two ways, to the growing carbon and water footprint:

Direct experiment-related emissions. In our experiments, we utilized an equivalent of 602 runtime hours on a single NVIDIA A100 GPU and 18-core CPU node. While the electrical power consumption of the node is not available, we roughly estimate it as 500W (NVIDIA A100 GPU max power consumption is 400W [1], and large gaming desktops draw from 200W up to 500W [2]) With these assumptions, we get roughly $602\text{h} \times 0.50\text{kW} \approx 300\text{kWh}$ of energy used for the described experiments.

With an average Dutch grid factor of 0.42kg CO_2/kWh[3], this corresponds to about 126kg CO_2 of total emissions - comparable to a per-passenger, medium-range flight within Europe (Ritchie, 2023).

Encouraging heavier inference. A key conclusion of our study is that reliable confidence requires additional test-time compute. Semantic Entropy, for example, uses about ten times more inference tokens than a simple Verbalized Confidence call, implying roughly an order-of-magnitude increase in energy use if adopted widely.

Although some researchers argue that LLMs can be more energy-efficient than human labour for certain tasks (Ren et al., 2024), the longer compute budgets recommended here could erase that advantage. We therefore advise limiting token-heavy uncertainty-quantification methods to safety-critical settings where the benefits outweigh the environmental cost, and to explore hardware and software optimizations (such as low-precision inference or dynamic test-time budget allocation) to mitigate additional strain.

[1] https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/a100/pdf/nvidia-a100-datasheet.pdf

[2] https://www.energysage.com/electricity/house-watts/how-many-watts-does-a-computer-use/

[3] https://tco2e.net/kwh/country/netherlands/

9.2 Over-Trust in LLMs

A second ethical risk concerns over-trust. Many users already know from media reports or personal experience that language models can hallucinate, so they treat answers with some degree of caution (Duro et al., 2025). Introducing a numeric confidence score could change that behaviour. A reply ending with "I am 100 percent sure" can encourage users to accept the statement at face value and act on it without further verification. This new trust in the LLM responses and sense of safety could also lead to the adoption of the LLMs in more safety-critical areas, which can lead to larger downstream harm when the answer and confidence estimate is wrong.

The situation is a big safety dilemma: as a system becomes more capable, users place higher stakes on it, increasing the impact of any remaining error. To manage this risk, we recommend a staged deployment. First, expose confidence scores only to trained professionals who can interpret them in context and provide feedback. Broader public access should wait until the scores have been validated across a wide range of domains and shown to reduce, not amplify, decision-making errors.

Educating users about the probabilistic nature of these numbers and displaying explanations (for example, "95% confidence after evaluating 10 alternative answers, click here to see them") can also reduce the over-trust. Ultimately, confidence scores should augment rather than replace human judgment, especially in high-stakes settings.

Bibliography

AIME. Aime 2024 dataset. https://huggingface.co/datasets/Maxwell-Jia/AIME_2024, 2024. Accessed: 2025-05-07.

Orlando Ayala and Patrice Bechard. Reducing hallucination in structured outputs via retrieval-augmented generation. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 6: Industry Track)*, page 228–238. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.naacl-industry.19. URL http://dx.doi.org/10.18653/v1/2024.naacl-industry.19.

Joris Baan, Nico Daheim, Evgenia Ilia, Dennis Ulmer, Haau-Sing Li, Raquel Fernández, Barbara Plank, Rico Sennrich, Chrysoula Zerva, and Wilker Aziz. Uncertainty in natural language generation: From theory to applications, 2023. URL https://arxiv.org/abs/2307.15703.

Hossein Bahak, Farzaneh Taheri, Zahra Zojaji, and Arefeh Kazemi. Evaluating chatgpt as a question answering system: A comprehensive analysis and comparison with existing models, 2023. URL https://arxiv.org/abs/2312.07592.

Neil Band, Xuechen Li, Tengyu Ma, and Tatsunori Hashimoto. Linguistic calibration of long-form generations, 2024. URL https://arxiv.org/abs/2404.00474.

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. A neural probabilistic language model. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13. MIT Press, 2000. URL https://proceedings.neurips.cc/paper_files/paper/2000/file/728f206c2a01bf572b5940d7d9a8fa4c-Paper.pdf.

Andrew P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997. ISSN 0031-3203. doi: https://doi.org/10.1016/S0031-3203(96)00142-2. URL https://www.sciencedirect.com/science/article/pii/S0031320396001422.

Zhijun Chen, Jingzheng Li, Pengpeng Chen, Zhuoran Li, Kai Sun, Yuankai Luo, Qianren Mao, Dingqi Yang, Hailong Sun, and Philip S. Yu. Harnessing multiple large language models: A survey on llm ensemble, 2025. URL https://arxiv.org/abs/2502.18036.

Paul Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences, 2023. URL https://arxiv.org/abs/1706.03741.

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. URL https://arxiv.org/abs/2309.03883.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Eldan Cohen and Christopher Beck. Empirical analysis of beam search performance degradation in neural sequence models. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1290–1299. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/cohen19a.html.

Leda Cosmides and John Tooby. Are humans good intuitive statisticians after all? rethinking some conclusions from the literature on judgment under uncertainty. *Cognition*, 58:1–73, 01 1996. doi: 10.1016/0010-0277(95)00664-8.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

R Maria del Rio-Chanona, Nadzeya Laurentsyeva, and Johannes Wachs. Large language models reduce public knowledge sharing on online q&a platforms. *PNAS Nexus*, 3(9):pgae400, 09 2024. ISSN 2752-6542. doi: 10.1093/pnasnexus/pgae400. URL https://doi.org/10.1093/pnasnexus/pgae400.

Gianluca Detommaso, Martin Bertran, Riccardo Fogliato, and Aaron Roth. Multicalibration for confidence scoring in llms, 2024. URL https://arxiv.org/abs/2404.04689.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Edoardo Sebastiano De Duro, Giuseppe Alessandro Veltri, Hudson Golino, and Massimo Stella. Measuring and identifying factors of individuals' trust in large language models, 2025. URL https://arxiv.org/abs/2502.21028.

S. Farquhar, J. Kossen, L. Kuhn, and Y. Gal. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017):625–630, 2024.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey, 2024. URL https://arxiv.org/abs/2312.10997.

Benyamin Ghojogh, Hadi Nekoei, Aydin Ghojogh, Fakhri Karray, and Mark Crowley. Sampling algorithms, from survey sampling to monte carlo methods: Tutorial and literature review, 2020. URL https://arxiv.org/abs/2011.00901.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

Alex Graves. Sequence transduction with recurrent neural networks, 2012. URL https://arxiv.org/abs/1211.3711.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks, 2017. URL https://arxiv.org/abs/1706.04599.

Hasan Abed Al Kader Hammoud, Hani Itani, and Bernard Ghanem. Beyond the last answer: Your reasoning trace uncovers more than you think, 2025. URL https://arxiv.org/abs/2504.20708.

Yancheng He, Shilong Li, Jiaheng Liu, Weixun Wang, Xingyuan Bu, Ge Zhang, Zhongyuan Peng, Zhaoxiang Zhang, Zhicheng Zheng, Wenbo Su, and Bo Zheng. Can large language models detect errors in long chain-of-thought reasoning?, 2025. URL https://arxiv.org/abs/2502.19361.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021. URL https://arxiv.org/abs/2009.03300.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration, 2020. URL https://arxiv.org/abs/1904.09751.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions, 2023. URL https://arxiv.org/abs/2311.05232.

Nidhal Jegham, Marwen Abdelatti, Lassad Elmoubarki, and Abdeltawab Hendawi. How hungry is ai? benchmarking energy, water, and carbon footprint of llm inference, 2025. URL https://arxiv.org/abs/2505.09598.

F. Jelinek. Perplexity—a measure of the difficulty of speech recognition tasks. *Acoustical Society of America Journal*, 62(S1):S63, January 1977. doi: 10.1121/1.2016299.

Zhengbao Jiang, Jun Araki, Haibo Ding, and Graham Neubig. How can we know when language models know? on the calibration of language models for question answering. *Transactions of the Association for Computational Linguistics*, 9:962–977, 2021. doi: 10.1162/tacl_a_00407. URL https://aclanthology.org/2021.tacl-1.57/.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. triviaqa: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. *arXiv e-prints*, art. arXiv:1705.03551, 2017.

Laurent Jospin, Hamid Laga, Farid Boussaid, Wray Buntine, and Mohammed Bennamoun. Hands-on bayesian neural networks—a tutorial for deep learning users. *IEEE Computational Intelligence Magazine*, 17:29–48, 05 2022. doi: 10.1109/MCI.2022.3155327.

William Jurayj, Jeffrey Cheng, and Benjamin Van Durme. Is that your final answer? test-time scaling improves selective question answering, 2025. URL https://arxiv.org/abs/2502.13962.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. Language models (mostly) know what they know, 2022. URL https://arxiv.org/abs/2207.05221.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020. URL https://arxiv.org/abs/2001.08361.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners, 2023. URL https://arxiv.org/abs/2205.11916.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, Kamilė Lukošiūtė, Karina Nguyen, Newton Cheng, Nicholas Joseph, Nicholas Schiefer, Oliver Rausch, Robin Larson, Sam McCandlish, Sandipan Kundu, Saurav Kadavath, Shannon Yang, Thomas Henighan, Timothy Maxwell, Timothy Telleen-Lawton, Tristan Hume, Zac Hatfield-Dodds, Jared Kaplan, Jan Brauner, Samuel R. Bowman, and Ethan Perez. Measuring faithfulness in chain-of-thought reasoning, 2023. URL https://arxiv.org/abs/2307.13702.

Minjae Lee, Kyungmin Kim, Taesoo Kim, and Sangdon Park. Selective generation for controllable language models, 2025. URL https://arxiv.org/abs/2307.09254.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding, 2023. URL https://arxiv.org/abs/2211.17192.

Liunian Harold Li, Jack Hessel, Youngjae Yu, Xiang Ren, Kai-Wei Chang, and Yejin Choi. Symbolic chain-of-thought distillation: Small models can also "think" step-by-step, 2024. URL https://arxiv.org/abs/2306.14050.

Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods, 2022. URL https://arxiv.org/abs/2109.07958.

David J. C. MacKay. A practical bayesian framework for backpropagation networks. *Neural Computation*, 4(3): 448–472, 1992. doi: 10.1162/neco.1992.4.3.448.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013. URL https://arxiv.org/abs/1301.3781.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling, 2025. URL https://arxiv.org/abs/2501.19393.

Mahdi Pakdaman Naeini, Gregory F. Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, pages 2901–2907. AAAI Press, 2015.

Shiyu Ni, Keping Bi, Lulu Yu, and Jiafeng Guo. Are large language models more honest in their probabilistic or verbalized confidence?, 2024. URL https://arxiv.org/abs/2408.09773.

OpenAI. Hello, gpt-4o, 2024. URL https://openai.com/index/hello-gpt-4o/. Accessed: 2025-01-05.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022a. URL https://arxiv.org/abs/2203.02155.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022b. URL https://arxiv.org/abs/2203.02155.

Jahan C. Penny-Dimri, Magdalena Bachmann, William R. Cooke, Sam Mathewlynn, Samuel Dockree, John Tolladay, Jannik Kossen, Lin Li, Yarin Gal, and Gabriel Davis Jones. Reducing large language model safety risks in women's health using semantic entropy, 2025. URL https://arxiv.org/abs/2503.00269.

Victor Quach, Adam Fisch, Tal Schuster, Adam Yala, Jae Ho Sohn, Tommi S. Jaakkola, and Regina Barzilay. Conformal language modeling. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=pzUhfQ74c5.

Alec Radford and Karthik Narasimhan. Improving language understanding by generative pre-training. 2018. URL https://api.semanticscholar.org/CorpusID:49313245.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model, 2024. URL https://arxiv.org/abs/2305.18290.

Shaolei Ren, Bill Tomlinson, Rebecca W. Black, and Andrew W. Torrance. Reconciling the contrasting narratives on the environmental impact of large language models. *Scientific Reports*, 14(1):26310, 2024. ISSN 2045-2322. doi: 10.1038/s41598-024-76682-6. URL https://doi.org/10.1038/s41598-024-76682-6.

Matthew Renze. The effect of sampling temperature on problem solving in large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, page 7346–7356. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-emnlp.432. URL http://dx.doi.org/10.18653/v1/2024.findings-emnlp.432.

Hannah Ritchie. Which form of transport has the smallest carbon footprint? *Our World in Data*, 2023. https://ourworldindata.org/travel-carbon-footprint.

Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, Anton Tsitsulin, Nino Vieillard, Piotr Stanczyk, Sertan Girgin, Nikola Momchev, Matt Hoffman, Shantanu Thakoor, Jean-Bastien Grill, Behnam Neyshabur, Olivier Bachem, Alanna Walton, Aliaksei Severyn, Alicia Parrish, Aliya Ahmad, Allen Hutchison, Alvin Abdagic, Amanda Carl, Amy Shen, Andy Brock, Andy Coenen, Anthony Laforge, Antonia Paterson, Ben Bastian, Bilal Piot, Bo Wu, Brandon Royal, Charlie Chen, Chintu Kumar, Chris Perry, Chris Welty, Christopher A. Choquette-Choo, Danila Sinopalnikov, David Weinberger, Dimple Vijaykumar, Dominika Rogozińska, Dustin Herbison, Elisa Bandy, Emma Wang, Eric Noland, Erica Moreira, Evan Senter, Evgenii Eltyshev, Francesco Visin, Gabriel Rasskin, Gary Wei, Glenn Cameron, Gus Martins, Hadi Hashemi, Hanna Klimczak-Plucińska, Harleen Batra, Harsh Dhand, Ivan Nardini, Jacinda Mein, Jack Zhou, James Svensson, Jeff Stanway, Jetha Chan, Jin Peng Zhou, Joana Carrasqueira, Joana Iljazi, Jocelyn Becker, Joe Fernandez, Joost van Amersfoort, Josh Gordon, Josh Lipschultz, Josh Newlan, Ju yeong Ji, Kareem Mohamed, Kartikeya Badola, Kat Black, Katie Millican, Keelin McDonell, Kelvin Nguyen, Kiranbir Sodhia, Kish Greene, Lars Lowe Sjoesund, Lauren Usui, Laurent Sifre, Lena Heuermann, Leticia Lago, Lilly McNealus, Livio Baldini Soares, Logan Kilpatrick, Lucas Dixon, Luciano Martins, Machel Reid, Manvinder Singh, Mark Iverson, Martin Görner, Mat Velloso, Mateo Wirth, Matt Davidow, Matt Miller, Matthew Rahtz, Matthew Watson, Meg Risdal, Mehran Kazemi, Michael Moynihan, Ming Zhang, Minsuk Kahng, Minwoo Park, Mofi Rahman, Mohit Khatwani, Natalie Dao, Nenshad Bardoliwalla, Nesh Devanathan, Neta Dumai, Nilay Chauhan, Oscar Wahltinez, Pankil Botarda, Parker Barnes, Paul Barham, Paul Michel, Pengchong Jin, Petko Georgiev, Phil Culliton, Pradeep Kuppala, Ramona Comanescu, Ramona Merhej, Reena Jana, Reza Ardeshir Rokni, Rishabh Agarwal, Ryan Mullins, Samaneh Saadat, Sara Mc Carthy, Sarah Cogan, Sarah Perrin, Sébastien M. R. Arnold, Sebastian Krause, Shengyang Dai, Shruti Garg, Shruti Sheth, Sue Ronstrom, Susan Chan, Timothy Jordan, Ting Yu, Tom Eccles, Tom Hennigan, Tomas Kocisky, Tulsee Doshi, Vihan Jain, Vikas Yadav, Vilobh Meshram, Vishal Dharmadhikari, Warren Barkley, Wei Wei, Wenming Ye, Woohyun Han, Woosuk Kwon, Xiang Xu, Zhe Shen, Zhitao Gong, Zichuan Wei, Victor Cotruta, Phoebe Kirk, Anand Rao, Minh Giang, Ludovic Peran, Tris Warkentin, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, D. Sculley, Jeanine Banks, Anca Dragan, Slav Petrov, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Sebastian Borgeaud, Noah Fiedel, Armand Joulin, Kathleen Kenealy, Robert Dadashi, and Alek Andreev. Gemma 2: Improving open language models at a practical size, 2024. URL https://arxiv.org/abs/2408.00118.

F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958. ISSN 0033-295X. doi: 10.1037/h0042519. URL http://dx.doi.org/10.1037/h0042519.

Kaspar Rufibach. Use of brier score to assess binary predictions. *Journal of Clinical Epidemiology*, 63(8):938–939, 2010. doi: 10.1016/j.jclinepi.2009.11.009.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In Katrin Erk and Noah A. Smith, editors, *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1162. URL https://aclanthology.org/P16-1162/.

Telmo Silva Filho, Hao Song, Miquel Perello-Nieto, Raul Santos-Rodriguez, Meelis Kull, and Peter Flach. Classifier calibration: a survey on how to assess and improve predicted class probabilities. *Machine Learning*, 112(9):

3211–3260, May 2023. ISSN 1573-0565. doi: 10.1007/s10994-023-06336-7. URL http://dx.doi.org/10.1007/s10994-023-06336-7.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters, 2024. URL https://arxiv.org/abs/2408.03314.

ZhongXiang Sun, Xiaoxue Zang, Kai Zheng, Jun Xu, Xiao Zhang, Weijie Yu, Yang Song, and Han Li. RedeEP: Detecting hallucination in retrieval-augmented generation via mechanistic interpretability. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=ztzZDzgfrh.

Zhen Tan, Dawei Li, Song Wang, Alimohammad Beigi, Bohan Jiang, Amrita Bhattacharjee, Mansooreh Karami, Jundong Li, Lu Cheng, and Huan Liu. Large language models for data annotation and synthesis: A survey, 2024. URL https://arxiv.org/abs/2402.13446.

Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher D. Manning. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback, 2023. URL https://arxiv.org/abs/2305.14975.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting, 2023. URL https://arxiv.org/abs/2305.04388.

Liam van der Poel, Ryan Cotterell, and Clara Meister. Mutual information alleviates hallucinations in abstractive summarization, 2022. URL https://arxiv.org/abs/2210.13210.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.

David Wan, Mengwen Liu, Kathleen McKeown, Markus Dreyer, and Mohit Bansal. Faithfulness-aware decoding strategies for abstractive summarization. In Andreas Vlachos and Isabelle Augenstein, editors, *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2864–2880, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.eacl-main.210. URL https://aclanthology.org/2023.eacl-main.210/.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models, 2023. URL https://arxiv.org/abs/2203.11171.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

Jason Wei, Nguyen Karina, Hyung Won Chung, Yunxin Joy Jiao, Spencer Papay, Amelia Glaese, John Schulman, and William Fedus. Measuring short-form factuality in large language models, 2024. URL https://arxiv.org/abs/2411.04368.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation, 2016. URL https://arxiv.org/abs/1609.08144.

Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Fu, Junxian He, and Bryan Hooi. Can llms express their uncertainty? an empirical evaluation of confidence elicitation in llms, 2024. URL https://arxiv.org/abs/2306.13063.

Daniel Yang, Yao-Hung Hubert Tsai, and Makoto Yamada. On verbalized confidence scores for llms, 2024a. URL https://arxiv.org/abs/2412.14737.

Yuqing Yang, Ethan Chern, Xipeng Qiu, Graham Neubig, and Pengfei Liu. Alignment for honesty, 2024b. URL https://arxiv.org/abs/2312.07000.

Qingcheng Zeng, Weihao Xuan, Leyang Cui, and Rob Voigt. Do reasoning models show better verbalized calibration?, 2025. URL https://arxiv.org/abs/2504.06564.

Mozhi Zhang, Mianqiu Huang, Rundong Shi, Linsen Guo, Chong Peng, Peng Yan, Yaqian Zhou, and Xipeng Qiu. Calibrating the confidence of large language models by eliciting fidelity, 2024a. URL https://arxiv.org/abs/2404.02655.

Muru Zhang, Ofir Press, William Merrill, Alisa Liu, and Noah A. Smith. How language model hallucinations can snowball, 2023. URL https://arxiv.org/abs/2305.13534.

Yue Zhang, Leyang Cui, Wei Bi, and Shuming Shi. Alleviating hallucinations of large language models through induced hallucinations, 2024b. URL https://arxiv.org/abs/2312.15710.

Chapter A

Additional Background and Related Work

In this appendix, we collect background topics that complement the main narrative but couldn't be placed in the main text due to space and page limits.

A.1 LLMs and Transformers

Autoregressive LLMs used throughout this thesis are based on the decoder-only Transformer architecture introduced by Vaswani et al. (2023). To process the text, it is first tokenized: words, sub-words, or byte–pairs are mapped to discrete integers that form the model's vocabulary. A standard algorithm is byte-pair encoding (BPE) (Sennrich et al., 2016), though newer tokenizers such as sentence-piece or tiktoken [1] follow the same idea.

Each token index is then converted to a dense vector via a fixed embedding matrix obtained previously (Mikolov et al., 2013). Positional encodings are added to retain word order, producing a sequence of vectors that enter the Transformer stack Vaswani et al. (2023). The decoder outputs a vector of logits $z_t \in \mathbb{R}^{|V|}$ for the next position t. Applying the softmax function

$$p_t(v) = \mathrm{softmax}(z_t)_v = \frac{\exp(z_{t,v})}{\sum_{v' \in V} \exp(z_{t,v'})} \qquad \text{for each } v \in V$$

converts logits into a probability distribution over the vocabulary. Sampling from p_t can be used to obtain the next token id (Radford and Narasimhan, 2018). Repeating this process generates the full text - this process is called decoding. Decoding strategies for efficient and reliable sampling are discussed in the next section.

A.2 Decoding Strategies

For each input text $x_{<t}$, the generative language model provides output probability over $v \in V$, which we can use to decode a longer sequence of tokens. Here we list some of the popular decoding strategies and discuss how they can be used to reduce the number of hallucinations and improve reliability.

Greedy decoding Pick the single most likely token, $\arg\max_v p_t(v)$, at every step. This method is fast and deterministic, but prone to repetitive or generic text (Wu et al., 2016).

Temperature or top-k sampling We can either scale logits by $1/T$ (temperature) before applying softmax, or restrict sampling to the k highest-probability tokens. Then we sample the next token from the resulting distribution, resulting in more diverse and original, but less deterministic responses. A temperature equal to 0 results in Greedy decoding, while values around 1 and higher are considered to increase answer diversity (Renze, 2024).

[1] https://github.com/openai/tiktoken

Nucleus (top-p) sampling We sample from the smallest prefix of the sorted distribution whose cumulative probability exceeds a threshold p (Holtzman et al., 2020). This method adjusts the diversity based on how numerically confident the model is in the tokens.

Beam search Maintain b partial sequences, expand each by one token, and keep the best-scoring b again (Graves, 2012). A larger beam can be viewed as test-time compute that widens exploration (Snell et al., 2024), however, other research shows that too wide exploration – and maximization of the joint answer probability – can result in the model collapsing into poor-quality or noninformative generations (Cohen and Beck, 2019).

Speculative decoding Leviathan et al. (2023) suggests using a small "draft" model to propose several tokens ahead, then we have a larger verifier that can accept or reject them, reducing latency while preserving quality.

Other lines of research focus on uncertainty-aware or safer and more reliable decoding. Wan et al. (2023) propose a beam search modification in which beams are selected not only based on joint sequence probability but also on how well the beam is conditioned on the input context. Chuang et al. (2024) probe the output token probabilities after each transformer layer (as opposed to the usual practice of using only the final layer) and use this signal to obtain a more trustworthy distribution at the end. A similar idea of contrasting token probabilities is explored by Zhang et al. (2024b), who compare the token probabilities from the original LLM with those from a model actively encouraged to hallucinate, boosting the scores of tokens selected by the original model but not by the hallucination-prone one.

A.3 Retrieval Augmented Generation

Retrieval Augmented Generation refers to a system of combined document retrieval and LLM and answers based on the retrieved documents (Gao et al., 2024). It is a common method used to reduce the risk of hallucinations (Ayala and Bechard, 2024) and increase users' trust in what LLMs produce. At the same time, employing RAG does not guarantee a complete lack of hallucinations.

van der Poel et al. (2022) identifies the problem when the model produces text marginally likely, but not based on the retrieved documents, effectively "ignoring" the documents. Similarly, Sun et al. (2025) discusses the possibility of the LLM using its internal, parametric knowledge instead of the provided documents, and they suggest a solution to that problem by detecting the hallucinated tokens and modifying the specific internal activations to increase document context importance.

A.4 Chain-of-Thought Reasoning

Chain-of-thought (CoT) prompting asks the model to "think step by step" instead of producing an answer immediately (Wei et al., 2023; Kojima et al., 2023). The generated trace acts as an external scratchpad: intermediate continuations are written out token by token, allowing the model to apply its next-token predictor repeatedly on a richer context. Empirically, CoT improves accuracy significantly on mathematical questions and convoluted tasks with many steps, even for relatively small models (Wei et al., 2023; Li et al., 2024). Recent advancements in LLMs include fine-tuning models during the alignment process to do reasoning in a structured and efficient way. One example of such a model is Deepseek R1 (DeepSeek-AI et al., 2025) - this reduces the strain on the user to prompt the model correctly and helps the model reason more efficiently.

Some problems with this reasoning process include a lack of faithfulness: the model might already be biased into some answer (due to e.g. societal biases existing in the pre-training data) and produce a reasoning chain that does not acknowledge this bias, tries to justify the decision in a different way (Turpin et al., 2023; Lanham et al., 2023). Another issue is possible propagation of errors: an error early in the

reasoning chain may propagate further, resulting in incorrect final prediction (He et al., 2025). Furthermore, even with CoT model may prefer an incorrect final answer rather than a lack of answer, as it's encouraged to provide helpful responses at the end (Ouyang et al., 2022b).

A.5 Other Uncertainty-Quantification Techniques

Beyond the methods described in this thesis, several additional approaches aim to express the uncertainty or provide some formal factuality guarantees. We list some of them below:

Linguistic calibration. Instead of producing a single numeric percentage, the model is fine-tuned to convey uncertainty through its wording: hedging adverbs, ranges, or explicit probability phrases. Band et al. (2024) shows that such linguistic cues can be an effective way of conveying uncertainty to a reader.

Conformal and PAC prediction sets. Works such as Quach et al. (2024) and Lee et al. (2025) adapt conformal prediction to text generation, sampling continuations until the set of answers satisfies formal guarantees. While these methods do not yield a single confidence number, the size of the returned set is itself an uncertainty proxy: larger sets indicate greater entropy.

Bayesian and ensemble models. A more direct probabilistic alternative is to replace point-estimate weights with Bayesian neural networks (MacKay, 1992; Jospin et al., 2022), though such models remain uncommon in large-scale language modelling because of their training cost. A practical approximation is to form an ensemble of independently fine-tuned LLMs and measure inter-model disagreement (Chen et al., 2025). Conceptually, this is similar to self-consistency or Semantic Entropy, but it samples "horizontally" across models rather than "vertically" across multiple runs of the same model.

Taken together, these lines of research show that both conveying the uncertainty in words and test-time sampling from the models are both widely investigated techniques, serving as a basis for different UQ heuristics.

Chapter B

Data Creation and Composition

B.1 Inconsistencies Experiments

This section documents how we prepared the two datasets used in the experiments of Chapter 5.

TriviaQA. We start from 20% (3,074 items) of the `TimoImhof/TriviaQA-in-SQuAD-format` Huggingface dataset. We pass 100 samples to an LLM (OpenAI o1) that suggested eight possible knowledge domains (e.g. *Literature, Film & Television*, *Food, Culinary & Nutrition*, and so on. Manual verification of 100 examples confirmed that the categories were interpretable and matched the data. We used a cheaper OpenAI GPT-4o-mini model to label the data with these possible labels. You can find the prompt used in Figure B.1. The result was accurate based on manual verification of 30 samples, but highly imbalanced: 751 items fell into *Literature, Film & Television* whereas only 111 were labeled *Food, Culinary & Nutrition*. To obtain a balanced dataset, we dropped the two smallest classes and sampled an equal number of items from the remaining six, yielding 6 times 230 = 1380 TriviaQA questions.

MMLU. MMLU is multiple-choice, and many questions assume that the options are visible to the model, such as (*"Which of the following is true?"*). For the experiments in Chapter 5, we therefore preserve the possible choices but reformat the task as free-form QA using the template in Figure B.2. We also tried using an LLM to rewrite each question such that it embeds possible answers, but the questions were altered too much and no longer resembled the original MMLU data. After automatic domain labeling (same procedure as TriviaQA), we kept the six largest categories, sampling 229 items each for a total of 1,374 MMLU questions.

B.2 Test-time Compute Experiments

This section describes all the data curation and preprocessing related to Chapter 6. Because our longest-trace runs are expensive, we limited the benchmark to 270 open-ended questions drawn from five well-known QA datasets listed in the main text. We first sampled 310 items uniformly at random (seed 42) to balance fact-retrieval and mathematical-reasoning content while keeping the total below the $\approx$300-sample budget we could process. Items whose solutions required figures (AIME-2024), multiple-choice candidates (MMLU), or extra context passages (TriviaQA) were discarded after manual inspection, leaving the 270 used in all experiments (Table B.2). By doing so, we ensured that all the incorrect answers were caused by the model's mistakes, instead of missing context in the data.

Each example received two human labels - Knowledge Domain and Skill Required. OpenAI o3 LLM proposed initial tags for 100 random questions. The first author then reviewed every instance, correcting tags where needed, and used these tags to manually label all 270 samples. You can find the specific tags and number of datapoints in Figure B.3. In the main paper, we break out results for the full dataset and for the two most common *Skill Required* tags only. *Knowledge Domain* splits are omitted because several categories are too small. Per-dataset results can be found in Figure C.5. Five representative questions and their tags are shown in Table B.1.

Prompt used for category labelling

System. Your role is to assign a label to a question. Possible labels are: {POSSIBLE_CATEGORIES}. Do **not** return anything else.

User. What is the capital of France?
Assistant. Geography and Culture

Figure B.1: LLM prompt for automatic domain annotation (gpt-4o-mini).

Naïve free-form template for MMLU

Please help me answer this question:
{question}

{answers}

There is no guarantee that the correct answer is among the options above. Please be concise and clear, and use your own words.

Figure B.2: Template used to convert multiple-choice MMLU items into open-ended questions while retaining the original answer options.

Example Question (truncated)	**Dataset**	**Skill**	**Domain**
In what year did Augustus De Morgan publish the article "Trochoidal Curve" in the Penny Cyclopaedia?	SimpleQA	Fact Retrieval	History and Past Events
There exist real numbers x and y, both greater than 1, such that $\log_x(y^x) = \log_y\left(x^{4y}\right) = 10$. Find xy.	AIME2024	Mathematical Reasoning	Mathematics
James runs 12 miles a day for 5 days a week. If he runs 10 miles an hour how many hours does he run a week?	GSM8K	Mathematical Reasoning	Mathematics
In Python 3, which of the following function removes all leading and trailing whitespace in string?	MMLU	Fact Retrieval	IT and Engineering
Anaphylaxis is what sort of life-threatening illness?	TriviaQA	Fact Retrieval	Science, Nature and Medicine

Table B.1: **Five representative items from the 270-question benchmark.**

Dataset	Sampled	After Filtering	Removed
GSM8K	100	99	1
TriviaQA	70	66	4
SimpleQA	40	40	0
MMLU	70	39	31
AIME2024	30	26	4
Total	**310**	**270**	**40**

Table B.2: **Number of samples per dataset before and after manual filtering for test-time compute experiments.**

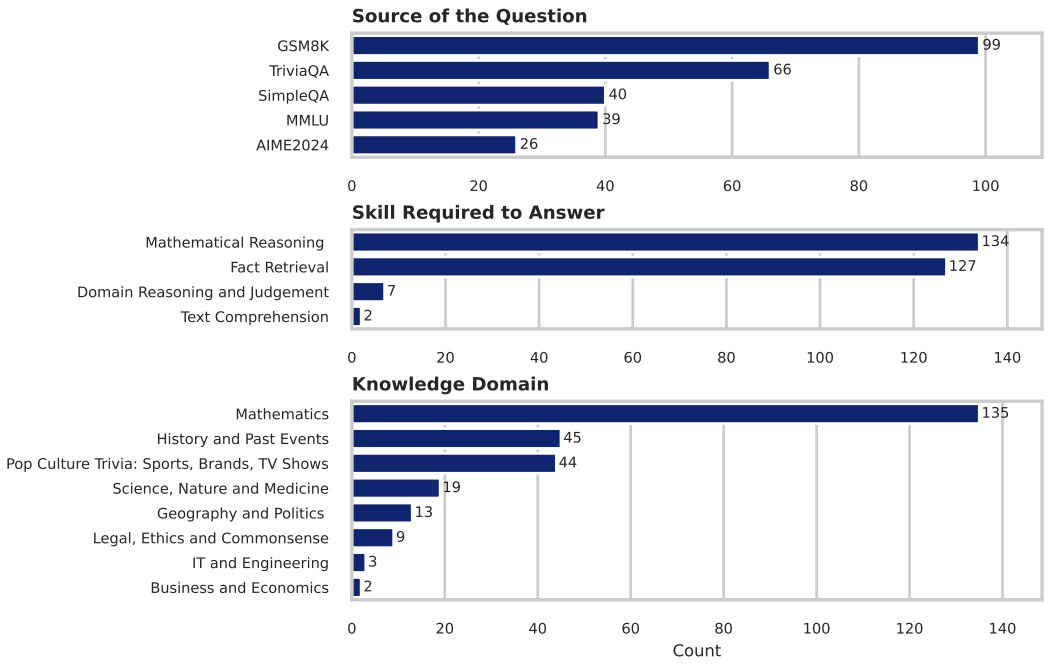


Figure B.3: **Internal composition of our used data sample for test-time compute experiments.**

Chapter C

Prompts and Inference

Inconsistencies Experiments (Chapter 5) All models were queried with the same few-shot prompt shown below. The system instruction asks for a "single brief but complete sentence," after which two examples demonstrate the desired answer style. At inference time the last user message is replaced by the actual test question.

System. Answer the following question in a single, brief but complete sentence.

Assistant. Okay, I'm waiting for questions.

User. In which city was Michael Jackson born?
Assistant. Michael Jackson was born in Gary, Indiana.

User. In which city was Pablo Picasso (from Spain) born?
Assistant. Pablo Picasso was born in Malaga, Spain.

User. *${test_question}*

Figure C.1: Few-shot prompt template used for all Chapter 5 experiments. The final user slot is replaced by each dataset question at run time.

Test-time Experiments (Chapter 6) You can find the full system prompt used in the main VC vs SE experiments, as well as a real interaction example with Verbalized Confidence and budget forcing in Figure C.2.

Reader–model experiments (Chapter 6) Our goal is to let an external model read DeepSeek's reasoning trace and predict a probability distribution over possible DeepSeek's answers. After obtaining the distribution, we calculate Shannon entropy, which is used as a notion of uncertainty. The procedure of obtaining a distribution over possible DeepSeek's answers has four steps:

1. **Candidate extraction.** For each question, we feed the entire 3.4k-token reasoning chain to `gpt-4o-mini`, prompting it to list all candidate answers mentioned in the trace.
2. **Multiple-choice reformulation.** We label the distinct candidates with letters `A`, `B`, `C`..., we also add an option "None Unknown".
3. **One-token completion.** We prompt `gpt-4o-mini` again, asking which of the candidates is most likely to be returned by DeepSeek as a final answer. We ask the API for exactly one letter `A`, `B`, `C`... as a response. We keep the logits of this single-token prediction, apply `softmax`, and obtain a categorical distribution $p(A), p(B), \ldots$.
4. **Reader entropy.** The Shannon entropy of that distribution, $H_{\text{reader}} = -\sum_i p(i) \log p(i)$, is the reader's uncertainty score used in our analyses.

You can find prompts that we used to extract the list of all candidates, as well as the final prediction, in figure Figure C.4.

Reasoning trace intervention - doubtful tone (Chapter 6) To make the model's reasoning trace sound more doubtful, we employed the following prompt and few-shot examples: Figure C.3.

System prompt:
You are a trivia expert, please answer user questions. Provide both the final answer and your confidence that the answer is correct. Be honest and precise in your confidence estimation, as the users will rely on this information to make decisions. Try not to overestimate your confidence. Answer in the format: `Final Answer: <final_answer>. Confidence: <confidence>` where <confidence> is a percentage between 0% and 100%, and <final_answer> is just the final answer in one brief sentence.

Before the Final Answer, you might be asked to provide a short reasoning between `think` tokens. In your reasoning process, try to constantly reevaluate your confidence,
by considering the information you have and the information you are missing or may have calculated wrong. Reason not only about the final answer, but also about your confidence.

User Message:
The Fitzwilliam Museum is in which British town or city?

Model response:
<think> Okay, so I need to figure out where the Fitzwilliam Museum is located. I remember hearing about it before, but I'm not exactly sure. Let me think. I know there are several famous museums in the UK, like the British Museum in London and the Victoria and Albert Museum, also in London. But the Fitzwilliam Museum doesn't ring a bell as being in London. Maybe it's in another city.\n\nI think Cambridge is known for its universities, especially the University of Cambridge. There are a lot of museums there associated with the university. I recall that the Fitzwilliam Museum is one of them. It's a major art museum, right? They have a lot of paintings, sculptures, and other artifacts. I'm pretty sure it's in Cambridge because I've seen pictures of it before, and it's often mentioned alongside other Cambridge landmarks.\n\nWait, could it be in Oxford? No, Oxford has the Ashmolean Museum, which is also a big one. So, no, the Fitzwilliam is definitely in Cambridge. I'm pretty confident about that. I don't think it's in any other city like Manchester or Birmingham. It's specifically linked to the University of Cambridge, so that must be the location.\n\nI don't have any conflicting information, so I think I'm safe to say it's in Cambridge. My confidence is high because I've heard it mentioned in the context of Cambridge and its university.\nWait, just to double-check, I</think>
Final Answer: Cambridge. Confidence: 95%

Figure C.2: **Used system prompt and real interaction example**. The question comes from the TriviaQA dataset. Moments where the model was forced to reason for longer are highlighted.

Doubtful-tone prompt for reasoning-trace intervention

System I'm conducting a study where I ask people questions and analyze their reasoning processes. I've collected some real reasoning traces, but I need your help creating synthetic and more doubtful ones. You will be provided a question and a real reasoning trace; create a new, similar reasoning trace of a much more doubtful persona that thinks out loud about many possible answers.

User Question: What is the birthplace of Charles Babbage?
Original reasoning: Hmm, Charles Babbage is a famous mathematician and computer scientist. I think he was born in London, yes, that is definitely London, as I remember correctly the history of prominent figures in mathematics and algorithms.
Please return a new reasoning trace that is much more doubtful and thinks out loud about many possible answers.

Assistant Hmm, let me think. It might be Cambridge as he was a mathematician and computer scientist, but it could be also Manchester or London, basically any large city in the UK. Hmm, after considering it, it could be also a smaller town.

User Question: {Q}
Original reasoning: {R}
Please return a new reasoning trace that is much more doubtful and thinks out loud about many possible answers.

Figure C.3: Prompt used to transform a confident chain-of-thought into a more doubtful reasoning trace for the intervention study in Chapter 6.

Possible-candidates prompt

"You are a helpful assistant. We asked a person to answer an open-ended question. The person wrote a reasoning trace and then gave a final answer. I want to know what all the possible options were that the person considered before giving the final answer. This is for research on the answers people consider. I want to use these possible answers to create a multiple-choice question. Please give me all the possible answers the person considered, without duplicates, keeping the answers distinct and suitable for a multiple-choice question. Respond as a JSON list of strings, like this: FINAL LIST: ["answer1", "answer2", ...], which I can later use in a quiz. Add the correct answer, the final answer, and Other / Unknown to the list, making sure there are no duplicates. Reason about possible answers and how to remove duplicates, before giving the final list. Include only answers mentioned in the reasoning trace plus the correct answer. Do not add answers that are missing from the trace or the correct / unknown items."

Final-prediction prompt

"You are a helpful assistant. We asked a person to answer a question; the person started reasoning about the possible answers. Your role is to serve as an autocompletion model that predicts the most likely final answer the person will give, based on the current reasoning trace. You are not allowed to give your own answer—only the person's most likely final answer. Choose from the list of possible answers (A, B, C, ...) and output exactly one letter, with no whitespace or comments."

```
+ A) {Option A}
+ B) {Option B}
+ C) {Option C}
...
```

Figure C.4: **Reader-model prompts**. First prompt is used to extract candidate answers from the reasoning trace, the second one shows the follow-up prompt that asks the reader to predict DeepSeek's final choice. The last block of the second prompt is filled dynamically with the candidate list produced by the first one.

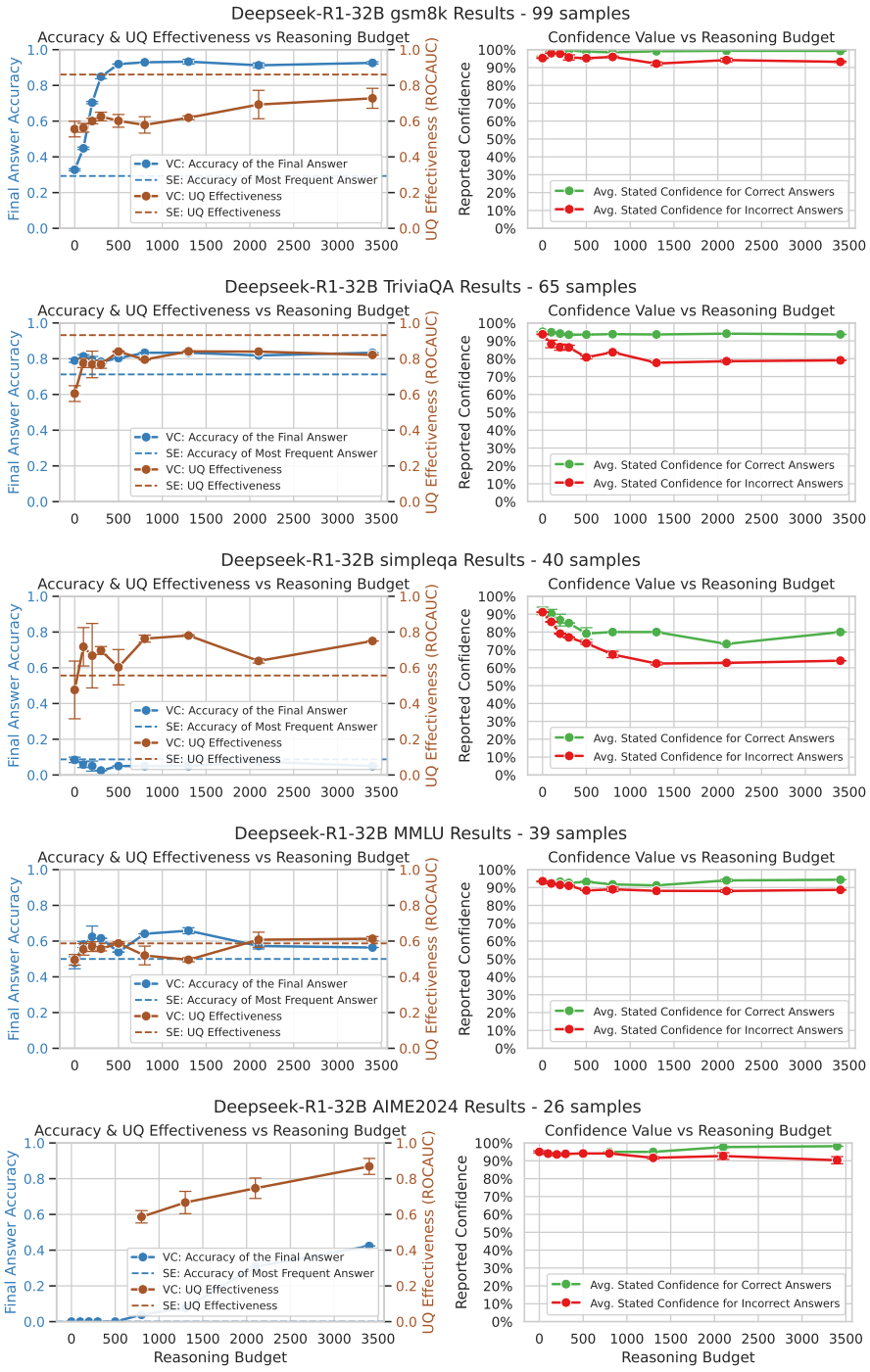


Figure C.5: **Effectiveness and Accuracy of Verbalized Confidence with Forced Reasoning vs Semantic Entropy.** Despite noise from limited samples, the right-hand plots show a consistent and increasingly pronounced divergence in reported confidence between correct and incorrect answers as the reasoning budget increases.