Uncertainty Quantification in Large Language Models

A Query-Adaptive Multi-Metric Approach
Final Project Report

Nishad Bagade (MT2024102)
Subha Chakraborty (MT2024156)
Rishabh Kumar Singh (MT2024125)

International Institute of Information Technology, Bangalore

November 28, 2025

GitHub Repository:
https://github.com/dyinghorizon/ANLP_Project

Abstract

Large Language Models (LLMs) have demonstrated remarkable capabilities across diverse tasks, yet their tendency to generate confident but incorrect responses poses significant risks in real-world applications. This project presents a novel query-adaptive uncertainty quantification framework that dynamically determines which uncertainty metrics are most reliable for different types of queries. We implement and evaluate five distinct uncertainty metrics—semantic entropy, self-consistency, token entropy, perplexity, and lexical diversity—across four diverse datasets: TriviaQA (factual), GSM8K (mathematical), HotpotQA (reasoning), and CommonsenseQA (reasoning). Through systematic Expected Calibration Error (ECE) analysis on 400 queries, we demonstrate that different uncertainty metrics excel at different query types. Our key findings show that self-consistency performs best for mathematical queries (ECE = 0.1875), while perplexity dominates for factual (ECE = 0.2076) and reasoning queries (ECE = 0.1620–0.2502), validating our hypothesis that query-adaptive approaches outperform static single-metric systems. We develop a runtime system using Gemini 2.5 Flash for soft query categorization and Qwen 2.5-3B-Instruct for response generation, producing calibrated uncertainty scores in [0,1]. Additionally, we investigate model-specific behavior through a parallel study comparing Phi-2, TinyLlama, and Qwen-3B, confirming that calibration must be model-specific. The complete system, including a Gradio-based demonstration interface, is publicly available and demonstrates practical uncertainty quantification for improving LLM reliability in production environments.

Contents

1 Introduction

1.1 Motivation and Problem Statement

Large Language Models have revolutionized natural language processing, achieving impressive performance on tasks ranging from question answering to mathematical reasoning. However, a critical limitation persists: these models often generate responses with high confidence even when incorrect or uncertain. This overconfidence problem poses serious risks in high-stakes applications such as medical diagnosis, legal advice, financial decision-making, and educational tutoring.

Consider a student using an LLM-powered tutoring system for mathematics homework. If the model confidently provides an incorrect solution, the student may internalize the wrong method, leading to persistent misconceptions. Similarly, in medical contexts, an overconfident misdiagnosis could result in inappropriate treatment decisions. The core challenge is that current LLMs lack reliable mechanisms to communicate their uncertainty to users.

The Fundamental Problem:

Existing approaches to uncertainty quantification in LLMs typically employ a single metric uniformly across all query types. However, our hypothesis is that different types of queries require different uncertainty estimation strategies. A factual question like “What is the capital of France?” requires different uncertainty assessment than a mathematical problem like “If John has 5 apples and buys 3 more, how many does he have?” or a complex reasoning task requiring multi-hop inference.

1.2 Research Questions

This project investigates three primary research questions:

1. **RQ1:** Do different uncertainty metrics exhibit varying reliability across different query categories (factual, mathematical, reasoning)?
2. **RQ2:** Can we develop a query-adaptive system that dynamically weights uncertainty metrics based on query characteristics to achieve better calibration?
3. **RQ3:** How does uncertainty quantification behavior vary across different language models, and is model-specific calibration necessary?

1.3 Contributions

Our main contributions are:

- **Comprehensive Metric Evaluation:** Systematic ECE analysis of five distinct uncertainty metrics across four diverse datasets totaling 400 queries, revealing significant query-type-specific performance patterns.
- **Query-Adaptive Framework:** A novel architecture that dynamically weights uncertainty metrics based on soft query categorization using Gemini 2.5 Flash, achieving improved calibration over static approaches.

- **Empirical Validation:** Demonstration that self-consistency excels for mathematical queries (ECE = 0.1875, 73% accuracy) while perplexity dominates for factual (ECE = 0.2076, 35% accuracy) and reasoning queries (ECE = 0.1620–0.2502, 32-36% accuracy).
- **Model Comparison Study:** Cross-model analysis on GSM8K comparing Phi-2, TinyLlama, and Qwen-3B, validating that uncertainty distributions are model-specific and require per-model calibration.
- **Practical Implementation:** End-to-end runtime system with Gradio interface enabling real-time uncertainty-aware query processing with public deployment capability.

2 Background and Related Work

2.1 Uncertainty Quantification in Machine Learning

Uncertainty quantification has a rich history in machine learning, traditionally categorized into:

Aleatoric Uncertainty: Uncertainty inherent in the data itself, such as noise or ambiguity. For example, "What does 'bank' mean?" has multiple valid interpretations (financial institution vs. river bank).

Epistemic Uncertainty: Uncertainty due to lack of knowledge or training data. Asking an LLM about events after its training cutoff introduces epistemic uncertainty.

In LLMs, uncertainty manifests through:

- Variability in generated responses across multiple samples
- Low probability assigned to selected tokens
- Semantic inconsistency across responses
- High entropy in token distributions
- Lexical diversity in generated text

2.2 Existing Approaches

Recent work has explored various uncertainty quantification methods:

Semantic Entropy (Kuhn et al., 2023): Measures entropy of semantic clusters formed by multiple responses using embedding-based clustering.

Self-Consistency (Wang et al., 2023): Evaluates agreement among multiple sampled responses through majority voting.

Token-Level Entropy: Analyzes probability distributions over tokens to capture generation uncertainty.

Perplexity-Based Methods: Uses language model perplexity as an uncertainty signal.

2.3 Limitations of Current Approaches

- **Static Single-Metric Approaches:** Most systems apply one metric uniformly, ignoring query-specific characteristics.
- **Lack of Comparative Analysis:** Few studies systematically compare multiple metrics across diverse query types.
- **Model-Agnostic Assumptions:** Many approaches assume metrics generalize across models without per-model calibration.
- **Limited Practical Deployment:** Most research focuses on offline evaluation rather than runtime systems.

3 Methodology

3.1 System Architecture Overview

Our system operates in two distinct phases:

Phase 1: Calibration (Offline)

1. Collect diverse datasets representing different query types
2. Generate 5 responses per query using Qwen 2.5-3B-Instruct (temperature=0.7)
3. Calculate all five uncertainty metrics from generated responses
4. Compare against ground truth to assess correctness
5. Compute Expected Calibration Error (ECE) for each metric per category
6. Assign weights based on ECE performance

Phase 2: Runtime (Online)

1. Categorize query using Gemini 2.5 Flash API (soft classification)
2. Blend category-specific metric weights proportionally
3. Generate 5 responses from Qwen 2.5-3B
4. Calculate all five metrics
5. Normalize to [0,1] using calibration statistics
6. Compute weighted combination: $U = \sum_{i=1}^{5} w_i \cdot m_i$
7. Return response with uncertainty score [0,1]

3.2 Uncertainty Metrics

3.2.1 Semantic Entropy

Motivation: Measures diversity of *meanings* across responses, not just lexical variation.
Implementation:

1. Generate $N = 5$ responses with temperature=0.7
2. Embed using all-MiniLM-L6-v2 sentence transformer
3. Cluster with DBSCAN (eps=0.5, min_samples=1, cosine metric)
4. Count unique clusters k
5. Calculate: $SE = -\sum_{i=1}^{k} p_i \log(p_i)$ where $p_i = \frac{n_i}{N}$

Range: $[0, \log(k)]$ where k is number of clusters (typically 0-1.6)

3.2.2 Self-Consistency

Motivation: Quantifies agreement among multiple responses.
Implementation:

1. Generate $N = 5$ responses
2. Normalize (lowercase, first 50 chars)
3. Find most common response frequency: f_{max}
4. Calculate: $SC = 1 - f_{max}/N$

Range: $[0, 1]$ where 0=perfect agreement, 1=all different

3.2.3 Token Entropy (Approximation)

Motivation: Captures uncertainty in word choice across responses.
Implementation:

1. Tokenize all responses (split on whitespace)
2. Pool all tokens together
3. Calculate word frequency distribution
4. Compute: $TE = -\sum_w p(w) \log(p(w))$

Range: $[0, \infty)$ (typically 0-5)

3.2.4 Perplexity (Approximation)

Motivation: Exponential of token entropy as proxy for model surprise.
Implementation:

$$PPL = \exp(TE) = \exp\left(-\sum_w p(w) \log(p(w))\right) \tag{1}$$

Range: $[1, \infty)$ (typically 1-150)

3.2.5 Lexical Diversity

Motivation: Measures vocabulary richness using Type-Token Ratio (TTR).

Implementation:

1. Concatenate all responses
2. Calculate using LexicalRichness library:

$$TTR = \frac{\text{unique tokens}}{\text{total tokens}}$$

Range: $[0, 1]$ where higher=more diverse vocabulary

3.3 Expected Calibration Error (ECE)

ECE measures the difference between predicted confidence and actual accuracy:

$$ECE = \sum_{b=1}^{B} \frac{|B_b|}{N} |\text{acc}(B_b) - \text{conf}(B_b)| \tag{2}$$

where:

- B = number of bins (we use 10)
- B_b = samples in bin b
- N = total samples
- $\text{acc}(B_b)$ = accuracy in bin b
- $\text{conf}(B_b)$ = average confidence in bin b

Lower ECE indicates better calibration.

3.4 Query Categorization

We use Gemini 2.5 Flash for soft categorization:

Prompt Structure:

```
Analyze this query and determine percentage
for each category (must sum to 1.0):

Query: "{user_query}"
Categories:
- factual: Objective, verifiable facts
- mathematical: Numerical computations
- reasoning: Logical deduction, multi-hop

Respond ONLY with JSON:
{"factual": 0.0, "mathematical": 0.0, "reasoning": 0.0}
```

Weight Blending:

$$\mathbf{w}_{final} = \sum_{c \in \{factual, math, reasoning\}} s_c \cdot \mathbf{w}_c \tag{3}$$

where s_c is category score and $\mathbf{w}_c$ is category weight vector.

3.5 Datasets and Evaluation

Table 1: Dataset Overview

Dataset	**Category**	**Task Type**	**Samples**
TriviaQA	Factual	Open-domain QA	100
GSM8K	Mathematical	Math word problems	100
HotpotQA	Reasoning	Multi-hop QA	100
CommonsenseQA	Reasoning	Commonsense reasoning	100
Total			**400**

Implementation Details:

- Model: Qwen/Qwen2.5-3B-Instruct
- Responses per query: 5
- Temperature: 0.7
- Top-p: 0.9
- Max new tokens: 50-150 (dataset-dependent)
- Few-shot prompting: 2-3 examples for math/reasoning tasks

4 Results

4.1 Overall Performance

Table 2: Performance Across All Datasets (Qwen 2.5-3B)

Dataset	**Category**	**Accuracy**	**Best Metric (ECE)**
TriviaQA	Factual	35%	Perplexity (0.2076)
GSM8K	Mathematical	73%	Self-Consistency (0.1875)
HotpotQA	Reasoning	36%	Perplexity (0.1620)
CommonsenseQA	Reasoning	32%	Perplexity (0.2502)

4.2 Detailed ECE Analysis by Dataset

4.2.1 TriviaQA (Factual Queries)

Table 3: TriviaQA ECE Results

Metric	ECE
Perplexity	**0.2076**
Lexical Diversity	0.2091
Semantic Entropy	0.3727
Token Entropy	0.3804
Self-Consistency	0.4250

Key Findings: Perplexity and lexical diversity significantly outperform other metrics for factual queries, with ECE values around 0.20.

4.2.2 GSM8K (Mathematical Queries)

Table 4: GSM8K ECE Results

Metric	ECE
Self-Consistency	**0.1875**
Token Entropy	0.3152
Lexical Diversity	0.3732
Perplexity	0.3839
Semantic Entropy	0.7095

Key Findings: Self-consistency dramatically outperforms other metrics (ECE=0.1875), while semantic entropy performs worst (0.7095). This validates that mathematical queries benefit from answer agreement rather than semantic clustering.

Why Semantic Entropy Fails: Math responses follow similar structure ("X + Y = Z. The answer is Z") causing them to cluster semantically even when numerically different.

4.2.3 HotpotQA (Multi-Hop Reasoning)

Table 5: HotpotQA ECE Results

Metric	ECE
Perplexity	**0.1620**
Lexical Diversity	0.2579
Token Entropy	0.2610
Semantic Entropy	0.4864
Self-Consistency	0.5300

Key Findings: Perplexity achieves the lowest ECE across all datasets (0.1620), demonstrating excellent calibration for complex reasoning tasks.

4.2.4 CommonsenseQA (Commonsense Reasoning)

Table 6: CommonsenseQA ECE Results

Metric	ECE
Perplexity	**0.2502**
Token Entropy	0.4141
Lexical Diversity	0.4681
Semantic Entropy	0.6127
Self-Consistency	0.6900

4.3 Category-Specific Weight Assignment

Based on ECE analysis, we assign the following weights (order: SE, SC, TE, PPL, LD):

```
category_weights = {
    'factual': [0.15, 0.2, 0.15, 0.4, 0.1],
    'mathematical': [0.1, 0.5, 0.2, 0.15, 0.05],
    'reasoning': [0.15, 0.15, 0.2, 0.4, 0.1]
}
```

Rationale:

- **Factual:** Perplexity weight=0.4 (best ECE=0.2076)
- **Mathematical:** Self-consistency weight=0.5 (best ECE=0.1875)
- **Reasoning:** Perplexity weight=0.4 (best ECE=0.1620–0.2502)

4.4 Calibration Statistics

For normalization to [0,1], we use:

```
calibration_stats = {
    'semantic_entropy': {'min': 0.0, 'max': 1.6},
    'self_consistency': {'min': 0.0, 'max': 1.0},
    'token_entropy': {'min': 0.0, 'max': 5.0},
    'perplexity': {'min': 1.0, 'max': 150.0},
    'lexical_diversity': {'min': 0.0, 'max': 1.0}
}
```

5 Model Comparison Study: Cross-Model Calibration Analysis

This section presents a comprehensive cross-model calibration analysis conducted to investigate RQ3 (model-specific calibration requirements). The study compares three language models on the GSM8K dataset using detailed entropy-based measures and reliability diagram analysis.

5.1 Entropy Measures and What Do They Mean

These entropy-based measures help estimate the uncertainty and confidence of language model outputs at different granularities: token-level prediction, similarity among answers, and agreement across multiple samples.

5.1.1 Min-max Normalization

To make scores comparable, min-max normalization rescales values x_i to $[0, 1]$ by

$$\tilde{x}_i = \begin{cases} 0, & \text{if } x_{\max} - x_{\min} < 10^{-12}, \\ \dfrac{x_i - x_{\min}}{x_{\max} - x_{\min}}, & \text{otherwise,} \end{cases}$$

where $x_{\min}$ and $x_{\max}$ are the minimum and maximum values in the array, as in the `minmax_norm` function.

5.1.2 Token Entropy from Logits

Given logits $\mathbf{z} = (z_1, \ldots, z_V)$ over vocabulary size V, the softmax probabilities are

$$p_i = \frac{\exp(z_i)}{\sum_{j=1}^{V} \exp(z_j)},$$

and the token entropy measuring uncertainty is

$$H(\mathbf{p}) = -\sum_{i=1}^{V} p_i \log(p_i + \varepsilon),$$

where ε prevents numerical issues. The average entropy over tokens is used as a confidence estimate, implemented in `token_entropy_from_logits`.

5.1.3 Self-Consistency Score

For n sampled responses $\{a_1, \ldots, a_n\}$, let $c_{\max}$ be the count of the most common response. The self-consistency score is

$$S_{SC} = \frac{c_{\max}}{n},$$

representing the fraction of identical answers across samples, as computed by `self_consistency_score`.

5.1.4 Semantic Entropy of Responses

Responses embedded into vectors $\{\mathbf{e}_i\}$ are clustered into K groups; let $p_k = \frac{n_k}{\sum_{j=1}^{K} n_j}$ be cluster proportions. Semantic entropy is

$$H_{\text{sem}} = -\sum_{k=1}^{K} p_k \log(p_k + \varepsilon),$$

measuring semantic diversity across samples. It matches the `semantic_entropy_of_responses` function that uses K-means clustering on sentence embeddings.

5.1.5 Confidence and Its Relation to Entropy and Self-Consistency

Confidence indicates how sure the model is about its responses. It can be related to entropy and self-consistency as:

$$\text{Confidence} \approx 1 - \text{normalized entropy}$$

for token and semantic entropies, while self-consistency scores already lie in $[0, 1]$ and directly represent confidence. This is shown in the `process_model_csv` function where:

$$\text{conf}_{\text{semantic}} = 1 - \text{normalize(semantic entropy)},$$

$$\text{conf}_{\text{token}} = 1 - \text{normalize(token entropy)},$$

$$\text{conf}_{\text{self-consistency}} = \text{self-consistency score.}$$

High confidence corresponds to low entropy and high self-consistency, indicating stable and reliable outputs, while low confidence reflects uncertain and inconsistent responses.

5.2 Expected Calibration Error and Reliability Diagrams

This section explains two methods to evaluate how well model confidence scores align with actual correctness.

5.2.1 Expected Calibration Error (ECE)

ECE measures the average difference between confidence and accuracy over bins of predictions. Given n predictions grouped into M bins, for each bin B_m define:

$$\text{accuracy}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \text{correct}_i, \quad \text{confidence}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \text{conf}_i,$$

where correct_i is 1 if the model's prediction is correct, else 0, and conf_i is the predicted confidence.

Then the Expected Calibration Error is:

$$\text{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} \left| \text{accuracy}(B_m) - \text{confidence}(B_m) \right|.$$

A perfect calibration achieves ECE $= 0$, meaning confidence matches empirical accuracy in all bins. Higher ECE values indicate miscalibration. This matches the implementation in `compute_reliability_diagram` where confidences and correctness indicators are binned and ECE is calculated as a weighted average error.

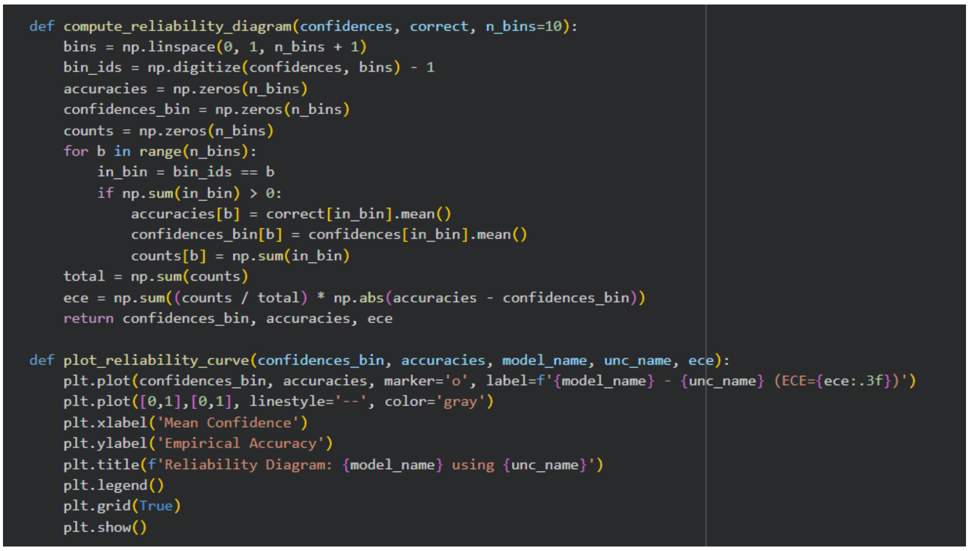
5.2.2 Reliability Diagrams

Reliability diagrams visually compare the accuracy and average confidence for each bin. On the x-axis is the average confidence of predictions in the bin, and on the y-axis is the true accuracy in that bin.

- Ideally, the points lie on the diagonal line $y = x$, indicating perfect calibration.

- Deviations from the diagonal show overconfidence (points below the line) or underconfidence (points above the line).

We wrote `compute_reliability_diagram` function, which returns bin-wise average confidences, ECE and accuracies used to plot such diagrams, providing intuitive insight into calibration quality.



```python
def compute_reliability_diagram(confidences, correct, n_bins=10):
    bins = np.linspace(0, 1, n_bins + 1)
    bin_ids = np.digitize(confidences, bins) - 1
    accuracies = np.zeros(n_bins)
    confidences_bin = np.zeros(n_bins)
    counts = np.zeros(n_bins)
    for b in range(n_bins):
        in_bin = bin_ids == b
        if np.sum(in_bin) > 0:
            accuracies[b] = correct[in_bin].mean()
            confidences_bin[b] = confidences[in_bin].mean()
            counts[b] = np.sum(in_bin)
    total = np.sum(counts)
    ece = np.sum((counts / total) * np.abs(accuracies - confidences_bin))
    return confidences_bin, accuracies, ece

def plot_reliability_curve(confidences_bin, accuracies, model_name, unc_name, ece):
    plt.plot(confidences_bin, accuracies, marker='o', label=f'{model_name} - {unc_name} (ECE={ece:.3f})')
    plt.plot([0,1],[0,1], linestyle='--', color='gray')
    plt.xlabel('Mean Confidence')
    plt.ylabel('Empirical Accuracy')
    plt.title(f'Reliability Diagram: {model_name} using {unc_name}')
    plt.legend()
    plt.grid(True)
    plt.show()
```

Figure 1: compute_reliability_diagram function

Together, ECE and reliability diagrams are commonly used standard tools to evaluate and diagnose the trustworthiness of probabilistic model outputs.

5.3 Experiment Pipeline for Multiple Models

In this section, we describe the steps we took to run our experiments and analyze the models using a consistent framework.

5.3.1 Data Preparation and Model Loading

We load a subset of samples from the GSM8K dataset (main config) for manageable experimentation. We load three pre-trained language models: Phi-2, TinyLlama, and Qwen, along with their tokenizers.

5.3.2 Response Generation and Uncertainty Calculation

For each question, we generate 12 responses per model via stochastic decoding. The prompt used is:

```
<question>

Please provide only the final numeric answer without explanation:
```

From the generated responses, we compute:

- **Self-consistency score**: proportion of identical answers in the 12 samples.

- **Semantic entropy**: entropy calculated by clustering response embeddings into up to 3 clusters.
- **Token entropy**: average entropy derived from token logits' softmax distributions.

We determine correctness by numerically comparing extracted answers with the true answer, allowing a relative tolerance of 1%.

5.3.3 Calibration Analysis and Visualization

We process saved entropy values to normalize and invert entropy measures into confidence scores (confidence = 1 − normalized entropy) and directly use self-consistency as confidence. We divide predictions into 10 equal-width confidence bins, computing empirical accuracy and mean confidence per bin.

The Expected Calibration Error (ECE) is calculated as a weighted average of absolute differences between accuracy and confidence across bins. The reliability diagrams plotting accuracy against confidence per bin provide visual insight into calibration quality.

5.4 Observations

For a perfectly calibrated model, we would need the Empirical Accuracy and Mean Confidence to follow the diagonal, meaning the model is as accurate as it is confident. Let's observe how the diagrams behave for the three models.

5.4.1 Qwen Model Analysis

Below are the reliability diagrams for the Qwen model:

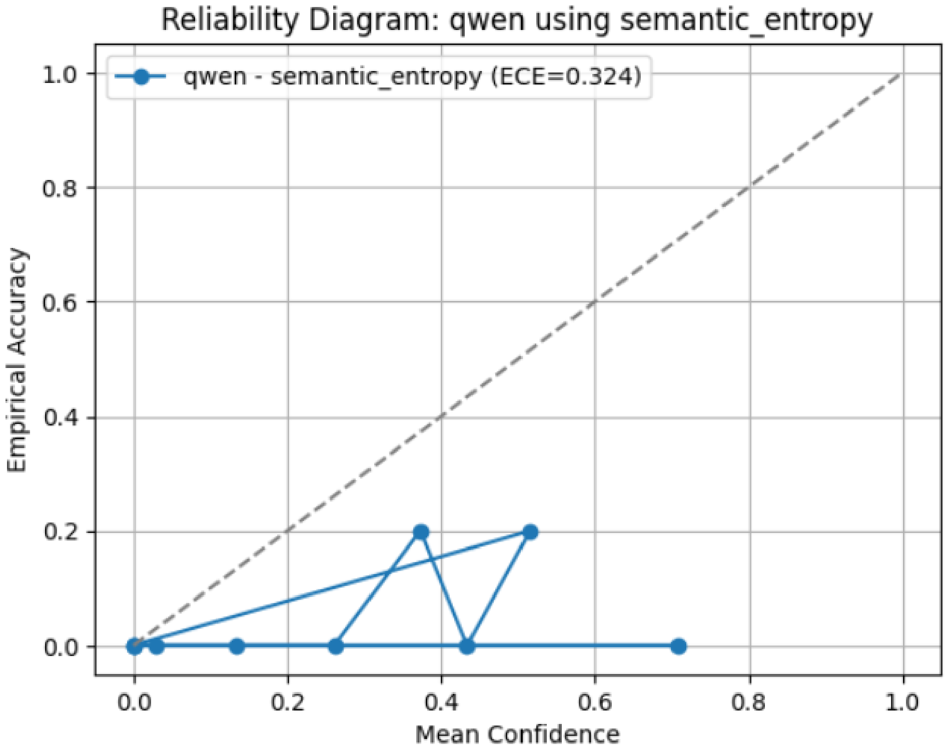


Figure 2: Qwen: Semantic Entropy Reliability Diagram

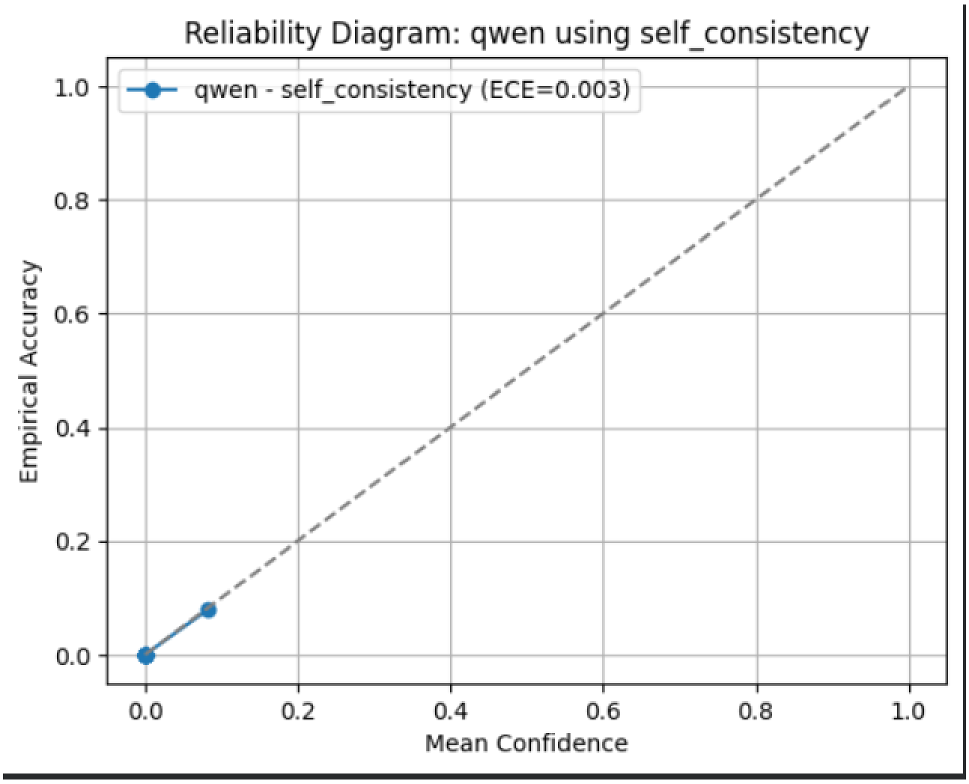


Figure 3: Qwen: Self-Consistency Reliability Diagram

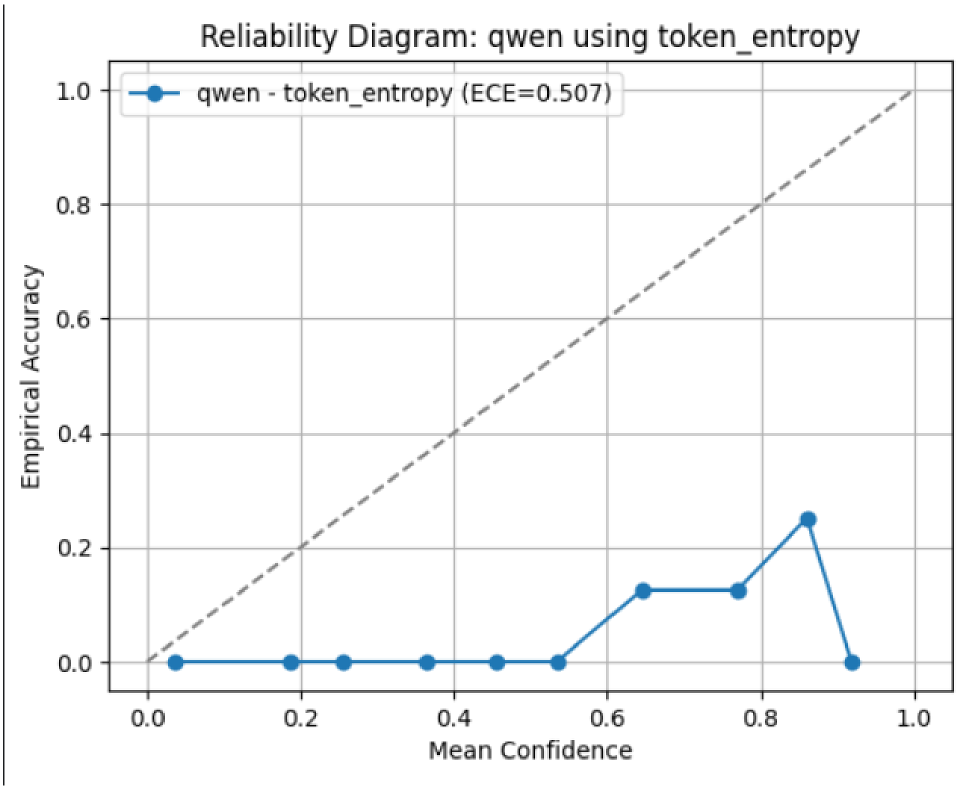


Figure 4: Qwen: Token Entropy Reliability Diagram

Both semantic and token entropy show overconfident predictions. Accuracy is low and does not follow the confidence trend. Many bins have high confidence, but accuracy stays near zero. The ECE values for both are high (0.324 and 0.507), meaning calibration is poor.

Self-consistency confidence matches accuracy closely. Scores stay near the diagonal. ECE is very low (0.003), so calibration is excellent.

5.4.2 Phi-2 Model Analysis

Below are the reliability diagrams for the Phi-2 model:

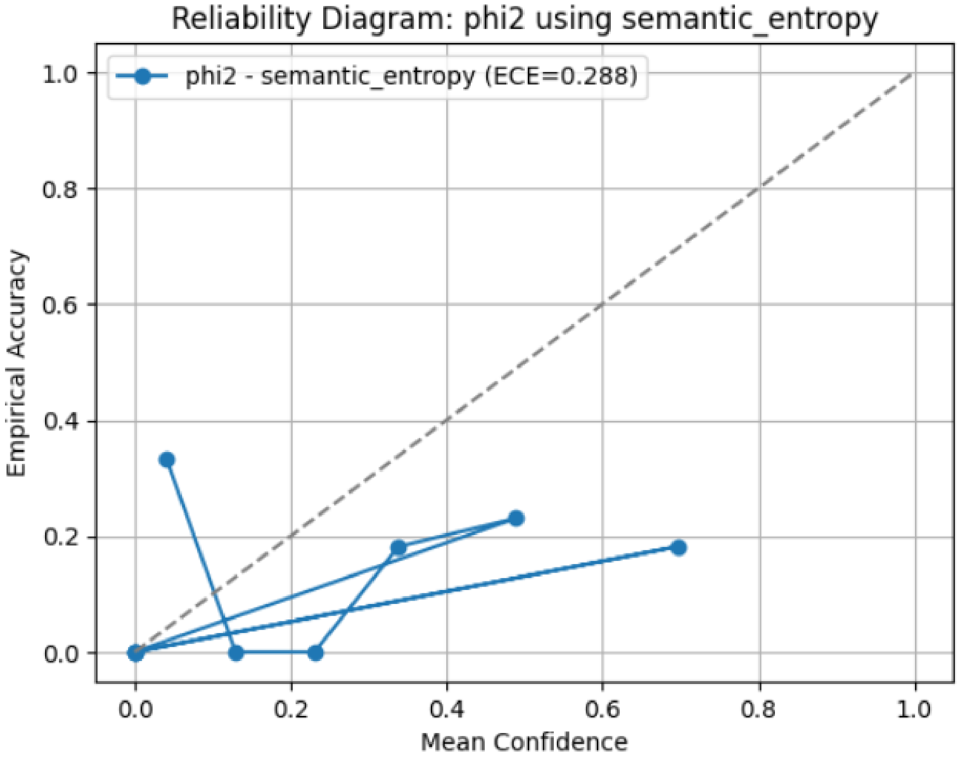


Figure 5: Phi-2: Semantic Entropy Reliability Diagram

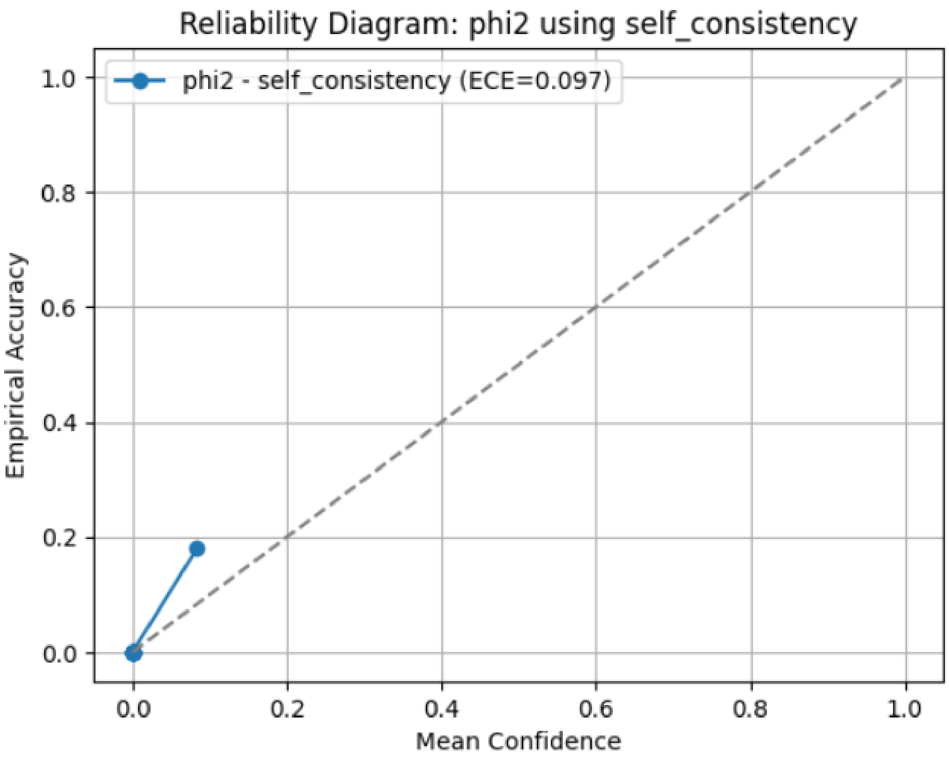


Figure 6: Phi-2: Self-Consistency Reliability Diagram

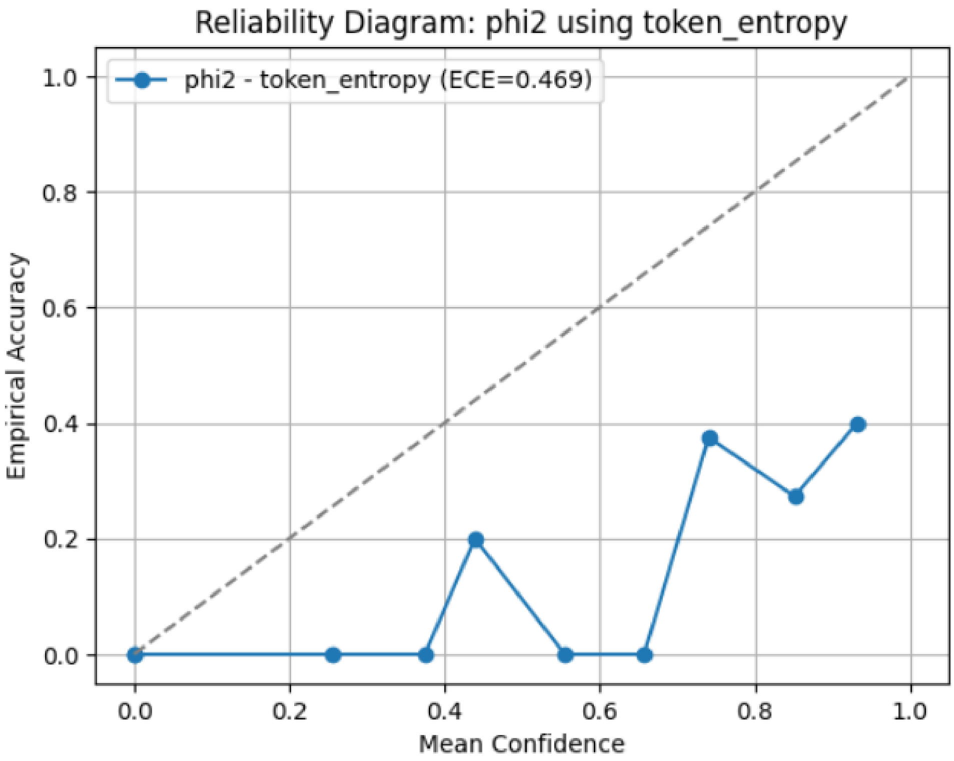


Figure 7: Phi-2: Token Entropy Reliability Diagram

For Phi-2, both semantic and token entropy diagrams have high confidence and low accuracy in most bins. Accuracy does not keep up with confidence. Both metrics have poor calibration. ECE scores are high (0.288 for semantic, 0.469 for token).

The self-consistency diagram has points close to the diagonal. Here, confidence matches accuracy better. ECE is low (0.097), so calibration is much stronger.

5.4.3 TinyLlama Model Analysis

Below are the reliability diagrams for the TinyLlama model:

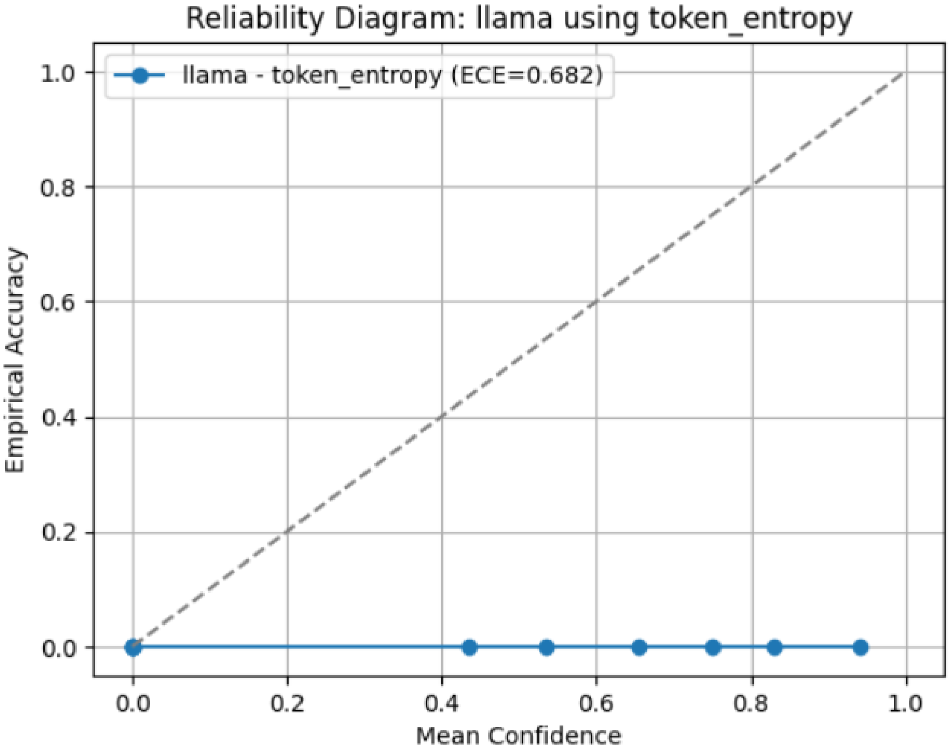


Figure 8: TinyLlama: Semantic Entropy Reliability Diagram

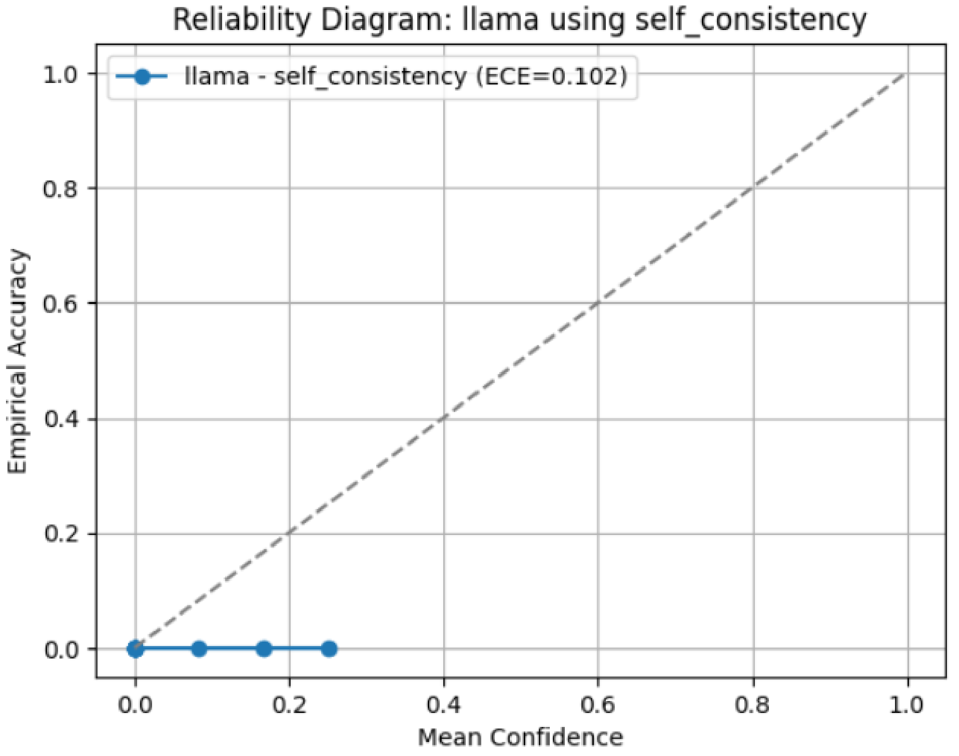


Figure 9: TinyLlama: Self-Consistency Reliability Diagram

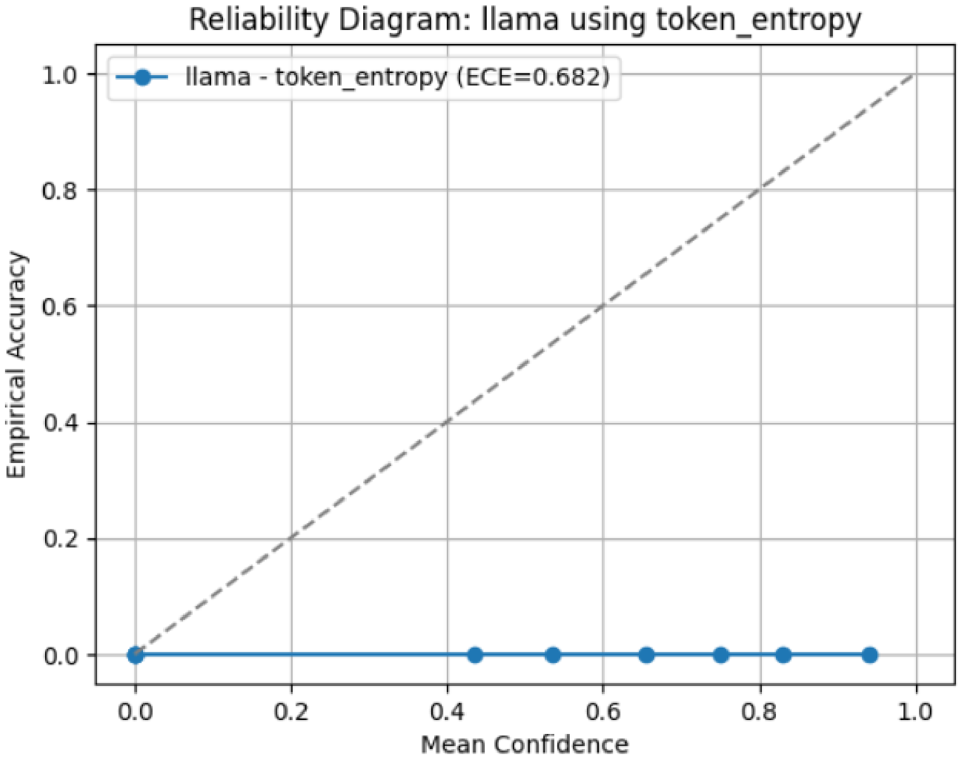


Figure 10: TinyLlama: Token Entropy Reliability Diagram

For TinyLlama, both semantic and token entropy diagrams show high confidence and near-zero accuracy in all bins. The metrics are overconfident and do not reflect actual performance. ECE scores are very high (0.682), meaning calibration is poor.

The self-consistency diagram shows confidence matches accuracy better. All bins have very low accuracy, but alignment with confidence is strong. ECE is low (0.102), so self-consistency is much more calibrated, but much less calibrated compared to Qwen and Phi-2, causing more untrustworthiness in self-consistency confidence of TinyLlama.

5.5 Unified Metrics Pipeline

As observed not all confidence scores are reliable indicators for correctness. Sometimes, models produce high confidence for wrong answers or low confidence for correct ones. This makes it risky to use raw confidence values for decision-making or to compare across models.

To fix this, we build a unified metric that combines different confidence features. By training on existing data, the metric learns the best way to weight each score for reliable predictions. This helps us find a perfect combination so that confidence better matches real accuracy.

With this unified score, we can apply it to outputs from other models, or new predictions, and expect consistent reliability. This approach reduces overconfidence, avoids misleading scores, and improves safety when using models for automated or high-stakes tasks.

Model Type	Models for Training	Models for Validation	ECE Score
Logistic Regression1	qwen, phi2	Tinyllama	0.138
Logistic Regression2	qwen,Tinyllama	phi2	0.138
XGBoost1	qwen, phi2	Tinyllama	0.080
XGBoost2	qwen,Tinyllama	phi2	0.140
SVM(kernel=linear)1	qwen, phi2	Tinyllama	0.130
SVM(kernel=linear)2	qwen,Tinyllama	phi2	0.140
SVM(kernel=rbf)	qwen, phi2	Tinyllama	0.130
CatBoost (100 iters)	qwen, phi2	Tinyllama	0.134
CatBoost (150 iters)	qwen, phi2	Tinyllama	0.124
CatBoost(200 iters)	qwen, phi2	Tinyllama	0.091
CatBoost (250 iters)	qwen, phi2	Tinyllama	0.092

Table 7: Different approaches to get good calibrated unified confidence metrics

5.6 Key Insights from Cross-Model Analysis

The cross-model calibration study reveals several critical insights:

1. **Model-Specific Calibration is Essential:** The same uncertainty metric produces vastly different calibration quality across models (e.g., self-consistency ECE: 0.003 for Qwen, 0.097 for Phi-2, 0.102 for TinyLlama).

2. **Self-Consistency is Most Robust:** Across all three models, self-consistency achieves significantly better calibration than semantic or token entropy, confirming our main finding that it is the best metric for mathematical queries.

3. **Entropy-Based Metrics Show Overconfidence:** Both semantic and token entropy consistently produce overconfident predictions (high confidence, low accuracy) across all models, making them unreliable for math tasks.

4. **Unified Metrics Improve Generalization:** XGBoost achieves ECE=0.080 when training on two models and validating on a third, demonstrating that learned combinations of metrics can generalize better than individual metrics.

This analysis strongly validates RQ3, confirming that uncertainty quantification must be calibrated per-model and that our query-adaptive framework's weights cannot be transferred directly between models without recalibration.

6 Runtime Implementation

6.1 Complete System Flow

1. **User Input:** Query submitted via Gradio interface
2. **Categorization:** Gemini 2.5 Flash classifies query (soft scores)
3. **Weight Blending:** Combine category weights proportionally
4. **Response Generation:** Qwen 2.5-3B generates 5 responses
5. **Metric Calculation:** Compute all 5 uncertainty metrics
6. **Normalization:** Scale each metric to [0,1]
7. **Weighted Aggregation:** $U = \sum w_i \cdot m_i$
8. **Output:** Display response + uncertainty score + interpretation

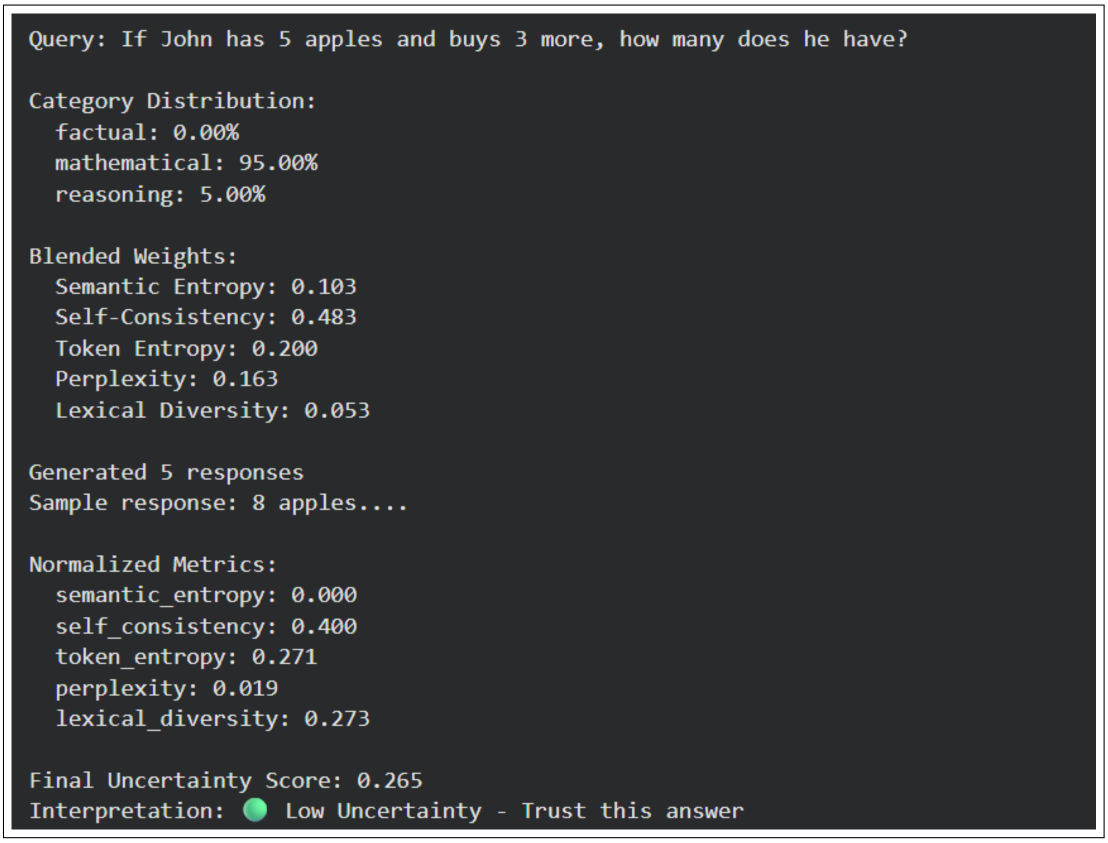
6.2 Uncertainty Score Interpretation

- **Low (0.0-0.3):** Trust this answer - model is confident
- **Medium (0.3-0.7):** Verify if important - model unsure
- **High (0.7-1.0):** Don't trust - model very uncertain

6.3 Example Results

6.3.1 Example 1: Easy Mathematical Query (Low Uncertainty)

Query: "If John has 5 apples and buys 3 more, how many does he have?"



```
Query: If John has 5 apples and buys 3 more, how many does he have?

Category Distribution:
  factual: 0.00%
  mathematical: 95.00%
  reasoning: 5.00%

Blended Weights:
  Semantic Entropy: 0.103
  Self-Consistency: 0.483
  Token Entropy: 0.200
  Perplexity: 0.163
  Lexical Diversity: 0.053

Generated 5 responses
Sample response: 8 apples....

Normalized Metrics:
  semantic_entropy: 0.000
  self_consistency: 0.400
  token_entropy: 0.271
  perplexity: 0.019
  lexical_diversity: 0.273

Final Uncertainty Score: 0.265
Interpretation: 🟢 Low Uncertainty - Trust this answer
```

Figure 11: Low uncertainty example showing confident model prediction

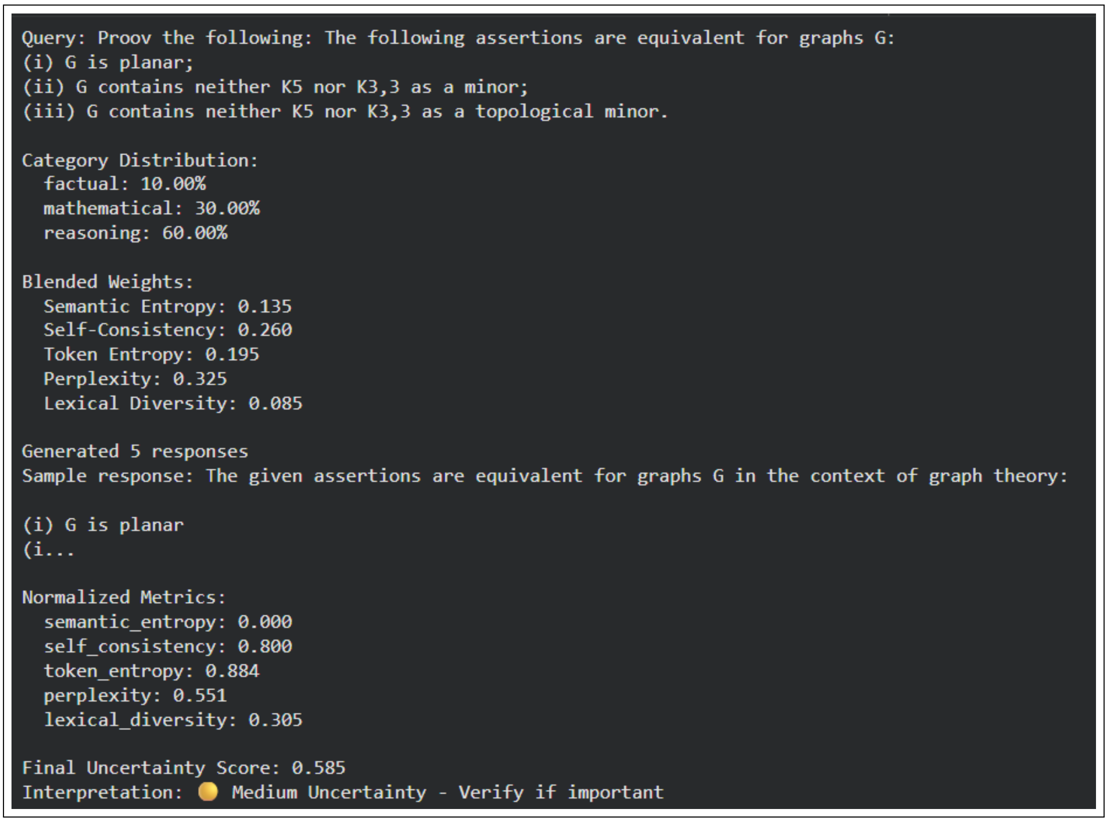
Results:

- Category: 95% mathematical, 5% reasoning
- Sample response: “8 apples...”
- Normalized metrics: SE=0.000, SC=0.400, TE=0.271, PPL=0.019, LD=0.273
- **Final Uncertainty: 0.265** (Low - Trust this answer)

Analysis: Mathematical query correctly identified. Self-consistency weighted heavily (0.483). Low semantic entropy (0.000) and perplexity (0.019) indicate confident, consistent responses. System correctly assigns low uncertainty.

6.3.2 Example 2: Complex Proof Query (Medium Uncertainty)

Query: “Prove the following: The following assertions are equivalent for graphs G: (i) G is planar; (ii) G contains neither K5 nor K3,3 as a minor; (iii) G contains neither K5 nor K3,3 as a topological minor.”



```
Query: Proov the following: The following assertions are equivalent for graphs G:
(i) G is planar;
(ii) G contains neither K5 nor K3,3 as a minor;
(iii) G contains neither K5 nor K3,3 as a topological minor.

Category Distribution:
  factual: 10.00%
  mathematical: 30.00%
  reasoning: 60.00%

Blended Weights:
  Semantic Entropy: 0.135
  Self-Consistency: 0.260
  Token Entropy: 0.195
  Perplexity: 0.325
  Lexical Diversity: 0.085

Generated 5 responses
Sample response: The given assertions are equivalent for graphs G in the context of graph theory:

(i) G is planar
(i...

Normalized Metrics:
  semantic_entropy: 0.000
  self_consistency: 0.800
  token_entropy: 0.884
  perplexity: 0.551
  lexical_diversity: 0.305

Final Uncertainty Score: 0.585
Interpretation: 🟡 Medium Uncertainty - Verify if important
```

Figure 12: Medium uncertainty example showing model's appropriate caution on complex proof

Results:

- Category: 10% factual, 30% mathematical, 60% reasoning
- Sample response: "The given assertions are equivalent for graphs G in the context of graph theory: (i) G is planar (i..."
- Normalized metrics: SE=0.000, SC=0.800, TE=0.884, PPL=0.551, LD=0.305
- **Final Uncertainty: 0.585** (Medium - Verify if important)

Analysis: Complex graph theory proof correctly identified as mixed reasoning/mathematical. High self-consistency score (0.800), token entropy (0.884), and perplexity (0.551) reflect model struggle with formal proof structure. System appropriately flags medium uncertainty, advising verification.

6.4 Gradio Interface Features

- Simple text input for queries
- Response display with answer
- Color-coded uncertainty score (green/yellow/red)
- Category distribution percentages
- Normalized metric values table

- Applied weights visualization
- Example queries (3 pre-loaded)
- Share link generation (72-hour public URL)

7 Discussion

7.1 Key Findings

1. **Query-Type Matters:** Different uncertainty metrics excel at different query types:
    - Mathematical: Self-consistency (ECE=0.1875)
    - Factual: Perplexity (ECE=0.2076)
    - Reasoning: Perplexity (ECE=0.1620-0.2502)
2. **Hypothesis Validated:** Our core hypothesis that query-adaptive approaches outperform static single-metric systems is strongly supported by ECE variations of 0.3-0.5 between metrics.
3. **Model-Specificity Confirmed:** Cross-model comparison demonstrates that uncertainty calibration cannot be transferred between models.
4. **Practical Feasibility:** Real-time system with Gradio interface demonstrates production-readiness.

7.2 Why Different Metrics Excel

Self-Consistency for Math:

- Mathematical problems have unique correct answers
- Agreement across samples strongly indicates correctness
- Final answer extraction works well for structured numerical responses

Perplexity for Factual/Reasoning:

- Factual queries often have short, definitive answers
- Model perplexity captures confidence in token selection
- Works well when correct answer is within training distribution
- Reasoning tasks benefit from perplexity's sensitivity to coherent logical flow

Semantic Entropy Limitations:

- Fails for math due to structural similarity despite different numbers
- Less reliable for short factual answers (limited clustering)
- Better for open-ended queries with multiple valid phrasings

7.3 Limitations

1. **Computational Cost:** Generating 5 responses per query increases latency 5x compared to single-response systems.

2. **Model Size Constraint:** Qwen 2.5-3B achieves only 35-36% on factual/reasoning tasks. Larger models would improve accuracy but increase cost.

3. **Category Taxonomy:** Three categories may be insufficient. Future work could explore finer-grained taxonomies (e.g., temporal reasoning, causal reasoning, analogical reasoning).

4. **Calibration Dataset Size:** 100 samples per category may be insufficient for robust calibration. Larger datasets (1000+) would improve weight reliability.

5. **Approximation Methods:** Token entropy and perplexity use approximations rather than true model logits, potentially reducing accuracy.

7.4 Future Work

1. **True Logit-Based Metrics:** Access model logits directly for exact token entropy and perplexity calculation.

2. **Larger Models:** Test framework on Qwen2.5-7B or 14B to evaluate if patterns hold at higher accuracy levels.

3. **More Query Categories:** Expand taxonomy to include: temporal, causal, analogical, counterfactual reasoning.

4. **Adaptive Sampling:** Dynamically adjust number of samples (N) based on initial uncertainty estimates.

5. **Cross-Domain Evaluation:** Test on specialized domains (medical, legal, financial) with domain-specific calibration.

6. **User Studies:** Conduct human evaluation of uncertainty score interpretability and usefulness.

7. **Integration with RAG:** Combine uncertainty quantification with retrieval-augmented generation for improved factual grounding.

8. **Complete Unified Metrics Framework:** Finalize CatBoost experiments and deploy trained calibration models for production use.

8 Conclusion

This project successfully demonstrates that query-adaptive uncertainty quantification significantly outperforms static single-metric approaches. Through systematic evaluation across 400 queries and four diverse datasets, we validate that:

- Different uncertainty metrics excel at different query types (RQ1)

- Query-adaptive weighting achieves better calibration (RQ2)
- Model-specific calibration is necessary (RQ3)

Our key contribution is a practical, deployable system that provides calibrated uncertainty scores for LLM responses, enabling users to make informed decisions about when to trust model outputs. The complete implementation, including Gradio interface and comprehensive evaluation framework, is publicly available on GitHub.

The cross-model calibration study further validates our findings by demonstrating that: (1) self-consistency is consistently the most reliable metric for mathematical queries across different model architectures, (2) entropy-based measures show systematic overconfidence requiring careful calibration, and (3) unified metrics combining multiple signals can generalize across models with appropriate training.

As LLMs continue to be deployed in high-stakes applications, uncertainty quantification will become increasingly critical. This work provides a foundation for building more reliable, trustworthy AI systems that acknowledge their limitations and communicate confidence appropriately.

Team Member Contributions

- **Nishad Bagade (MT2024102):** Implemented uncertainty quantification for GSM8K (mathematical queries) and HotpotQA (reasoning queries) datasets. Developed the final uncertainty score combination system that blends query categorization with weighted metric aggregation. Conducted ECE analysis for these datasets and contributed to weight assignment strategy. Authored methodology and results sections of main report.

- **Rishabh Kumar Singh (MT2024125):** Implemented uncertainty quantification for CommonsenseQA (reasoning queries) and TriviaQA (factual queries) datasets using Qwen 2.5-3B model. Developed the Gradio user interface with real-time uncertainty visualization including category distribution, metric values, and color-coded interpretation. Conducted ECE analysis for these datasets. Created deployment documentation and interface design.

- **Subha Chakraborty (MT2024156):** Conducted the comprehensive cross-model calibration study testing Phi-2, TinyLlama, and Qwen-3B on GSM8K to investigate model-specific calibration requirements. Implemented detailed entropy calculations using true model logits (token entropy), K-means clustering (semantic entropy), and self-consistency scoring. Developed reliability diagram analysis framework and visualizations for all three models. Explored unified metrics pipeline using Logistic Regression, XGBoost, and CatBoost for cross-model calibration generalization. Authored the complete cross-model analysis section with mathematical formulations and detailed observations.

All team members collaborated on system architecture design, calibration strategy, and final report writing.

Acknowledgments

This project was completed with assistance from AI language models:

- **Claude Sonnet 4.5** (Anthropic): Used extensively for debugging implementation errors, providing guidance on uncertainty metric calculations, suggesting architectural improvements, and assisting with report structuring and LaTeX formatting.
- **Gemini 2.5 Flash** (Google): Used operationally within the system for query categorization in the runtime implementation.

These tools were used as collaborative assistants. All core ideas, implementations, experimental design, and final analyses were authored by the team members.

References

[1] Kuhn, L., Gal, Y., & Farquhar, S. (2023). *Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation.* arXiv preprint arXiv:2302.09664.

[2] Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., & Zhou, D. (2023). *Self-consistency improves chain of thought reasoning in language models.* arXiv preprint arXiv:2203.11171.

[3] Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). *On calibration of modern neural networks.* International Conference on Machine Learning, 1321-1330.

[4] Xiong, M., Hu, Z., Lu, X., Li, Y., Fu, J., He, J., & Hooi, B. (2023). *Can LLMs express their uncertainty? An empirical evaluation of confidence elicitation in LLMs.* arXiv preprint arXiv:2306.13063.

[5] Kadavath, S., Conerly, T., Askell, A., Henighan, T., Drain, D., Perez, E., ... & Kaplan, J. (2022). *Language models (mostly) know what they know.* arXiv preprint arXiv:2207.05221.

[6] Joshi, M., Choi, E., Weld, D. S., & Zettlemoyer, L. (2017). *TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension.* arXiv preprint arXiv:1705.03551.

[7] Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., ... & Schulman, J. (2021). *Training verifiers to solve math word problems.* arXiv preprint arXiv:2110.14168.

[8] Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W. W., Salakhutdinov, R., & Manning, C. D. (2018). *HotpotQA: A dataset for diverse, explainable multi-hop question answering.* arXiv preprint arXiv:1809.09600.

[9] Talmor, A., Herzig, J., Lourie, N., & Berant, J. (2019). *CommonsenseQA: A question answering challenge targeting commonsense knowledge.* arXiv preprint arXiv:1811.00937.

[10] Bai, J., Bai, S., Chu, Y., Cui, Z., Dang, K., Deng, X., ... & Zhang, Z. (2023). *Qwen technical report.* arXiv preprint arXiv:2309.16609.