

---

# CLOTHO: MEASURING TASK-SPECIFIC PRE-GENERATION TEST ADEQUACY FOR LLM INPUTS

---

A PREPRINT

Juyeon Yoon<sup>1</sup>, Somin Kim<sup>1</sup>, Robert Feldt<sup>2</sup>, and Shin Yoo<sup>1</sup>

<sup>1</sup>KAIST, Daejeon, Republic of Korea

<sup>2</sup>Chalmers University of Technology, Gothenburg, Sweden

{juyeon.yoon, somin.kim, shin.yoo}@kaist.ac.kr  
robert.feldt@chalmers.se

September 25, 2025

## ABSTRACT

Software increasingly relies on the emergent capabilities of Large Language Models (LLMs), from natural language understanding to program analysis and generation. Yet testing them on specific tasks remains difficult and costly: many prompts lack ground truth, forcing reliance on human judgment, while existing uncertainty and adequacy measures typically require full inference. A key challenge is to assess input adequacy in a way that reflects the demands of the task, ideally before even generating any output. We introduce CLOTHO<sup>1</sup>, a task-specific, pre-generation adequacy measure that estimates input difficulty directly from hidden LLM states. Given a large pool of unlabelled inputs for a specific task, CLOTHO uses a Gaussian Mixture Model (GMM) to adaptively sample the most informative cases for human labelling. Based on this reference set the GMM can then rank unseen inputs by their likelihood of failure. In our empirical evaluation across eight benchmark tasks and three open-weight LLMs, CLOTHO can predict failures with a ROC-AUC of 0.716, after labelling reference sets that are on average only 5.4% of inputs. It does so without generating any outputs, thereby reducing costs compared to existing uncertainty measures. Comparison of CLOTHO and post-generation uncertainty measures shows that the two approaches complement each other. Crucially, we show that adequacy scores learnt from open-weight LLMs transfer effectively to proprietary models, extending the applicability of the approach. When prioritising test inputs for proprietary models, CLOTHO increases the average number of failing inputs from 18.7 to 42.5 out of 100, compared to random prioritisation.

## 1 Introduction

The emergent capabilities of Large Language Models (LLMs) have enabled new types of software systems, from domain-specific natural language understanding Kim et al. (2023) to autonomous coding and testing agents Zhang et al. (2024); Yang et al. (2024); Yoon et al. (2024); Rondon et al. (2025). As LLMs move beyond text generation into specific inference and reasoning tasks and see more widespread adoption, testing them becomes more critical and more costly. Yet the very qualities that make LLMs powerful also make them difficult to validate: their inner workings are opaque, and for many tasks we rely on them to solve, no ground truth is immediately available. Our goal is to improve task-specific LLM testing by *prioritising which inputs to run and label first*, ideally exposing failures earlier and with less effort.

Prior work on test adequacy in Deep Neural Networks (DNNs) have focused on structural coverage Pei et al. (2017); Ma et al. (2018) and distributional deviation from training data Kim et al. (2019); Kim and Yoo (2021). These test

---

<sup>1</sup>Clotho, the Greek goddess of fate, spins the thread of human life. The name suggests that our technique operates at the start of LLM inference, the pre-generation stage.

adequacy metrics enable us to prioritise manual labelling for new inputs Kim et al. (2020), or to guide automated input generation Tian et al. (2018). However, these approaches do not translate well to LLMs. First, LLM pre-training corpora are vast and heterogeneous Gao et al. (2020); Laurençon et al. (2023), making it infeasible to assess coverage or distributional density. Second, what matters in practice is not pre-training behaviour, it is rather how the model performs on a specific task as described by a fixed prompt<sup>2</sup>.

Since adequacy cannot be judged against pre-training data, we can turn to signals derived from the model itself. Transformer-based LLMs Vaswani et al. (2017) encode input as embeddings, process it through stacked attention and feed-forward layers, and decode responses token by token. Within this structure, several post-hoc uncertainty measures have been proposed: semantic entropy to detect hallucinations Farquhar et al. (2024), self-consistency as a proxy for reliability Wang et al. (2023), and measures like token-level confidence and inference variability to flag input risk Huang et al. (2025). While these metrics are informative, they are all *post-generation*. They exploit signals available only after output is produced, but thus require running full inference, often multiple times due to output non-determinism, which increases both computation time and cost. This limits their practicality for large-scale testing.

To tackle these challenges, we introduce CLOTHO, a task-scoped adequacy measure designed around two key elements. First, it embeds testing in an *active learning loop* Ren et al. (2021) that incrementally expands a small seed set of labelled inputs, reducing reliance on costly ground truth by focusing human effort where it matters most. Second, it operates *pre-generation*, using only input embeddings of an LLM to estimate adequacy before any output is produced. This avoids the overhead of full inference while still identifying inputs likely to reveal failures.

Concretely, CLOTHO constructs a Gaussian Mixture Model over the Last-token Input Hidden States (LIHS) of passing inputs in the labelled seed set. This surrogate model (i.e., predicting uncertainty in place of the target LLM) ranks unseen inputs by their likelihood-based “surprise”, in the spirit of Surprise Adequacy Kim et al. (2019, 2022), with low-likelihood inputs treated as more challenging. The reference set is expanded iteratively: new labelling targets are chosen where the existing distribution is ambiguous, and also where they add diversity, so that human effort focuses on the most informative cases. To align with reference set expansion, the GMM adapts its number of components and dimensionality guided by cross-validation and estimates of active component usage. While pre-generation signals require hidden states, we show that scoring models trained on small open-weight LLMs transfer effectively to closed-weight models (e.g., GPT, Claude, Gemini), enabling cost-effective testing even with API-only access.

We validate CLOTHO across eight benchmark tasks and three open-weight LLMs (Llama, Mistral, Gemma). After running and labelling just 5.4% of inputs, it achieves a ROC-AUC of 0.716 in predicting correctness—on par with post-generation uncertainty metrics that require repeated inferences. When used to prioritise test inputs for proprietary models, CLOTHO surfaces 42.5 failures out of the top 100 inputs on average, compared to 18.7 of random selection, an increase of 126.8%.

Our contributions are:

- We propose CLOTHO, a task-specific, pre-generation adequacy measure that ranks inputs based on learnt density over hidden LLM states. CLOTHO adaptively constructs a reference set of labelled inputs, minimising the human labelling efforts. The reference set is used to model the distribution of passing inputs in the input-side activation space, which in turn allows us to predict whether an unseen input will produce a correct output.
- We conduct an empirical study across eight tasks and three open-weight LLMs, showing strong prediction and prioritisation using limited labelling.
- We demonstrate complementarity with post-generation uncertainty measures used for hallucination detection, such as internal hidden state variance and semantic entropy.
- We show how adequacy scores learnt on small models transfer to proprietary LLMs, enabling cost-effective testing even with API-only access.

Section 2 motivates and validates our pre-generation, task-scoped premise with an exploratory analysis of Transformer hidden states. Section 3 then describes CLOTHO in detail. Section 4 lays out the details of our experimental setup, the result of which is presented in Section 5. Section 6 discusses further research directions, and Section 7 lists threats to validity. Section 8 presents related work and finally Section 9 concludes.

<sup>2</sup>Technically a prompt is a prompt template describing a task, into which specific inputs are then inserted.

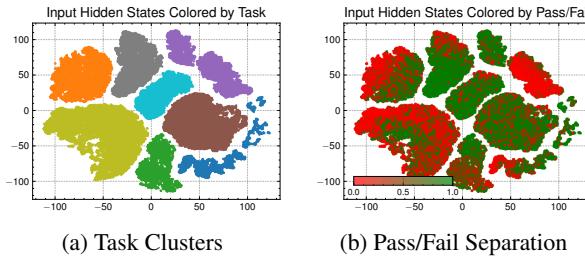


Figure 1: Visualisations on Llama 8B internal state space with inputs for all studied tasks

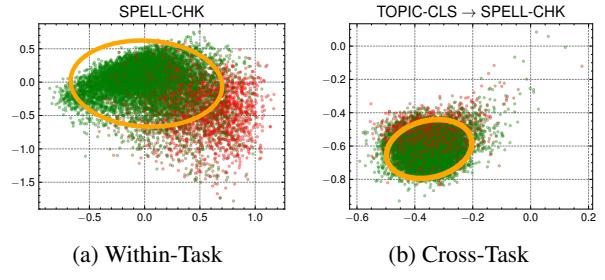


Figure 2: Pass/Fail Separation of SPELL-CHK Inputs in Projected PCA Space Fitted on Passing Inputs

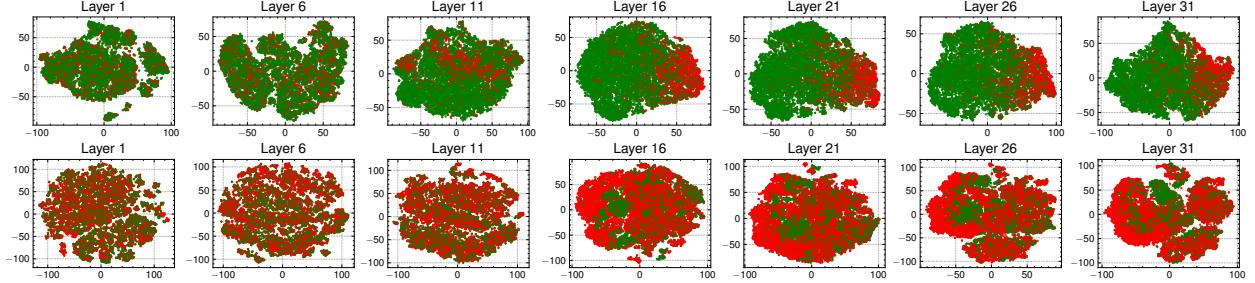


Figure 3: Input hidden states of Llama 8B from two prompted tasks (one per line) for varying depths of layers.

## 2 Exploratory Study: Transformer Hidden States under Task-specific Prompting

CLOTHO’s core assumption is that both the type of task and the relative difficulty of each input for a task are reflected in the internal hidden states within the Transformer architecture, even *before* any output token is produced. To verify the validity of these assumptions, several key questions need to be answered: (1) How do the Transformer internal states represent different tasks when the model is prompted with specific instructions? (2) Can we reliably distinguish potentially failing inputs from straightforward passing inputs based solely on Transformer internal states? (3) Which layers of the Transformer architecture provide the most useful hint for characterising input difficulty? In this section, to explore the feasibility of leveraging hidden states to derive test adequacy metrics, we present preliminary studies that aim to answer these questions.

### 2.1 Modelling Representative Cases in Task-specific Internal State Subspace

Figures 1a and 1b plot activation vectors from a hidden layer of Llama3.1 8B across inputs from eight natural and programming language tasks studied in this paper (see Section 4.2). We reduced the vectors to two dimensions with t-SNE Maaten and Hinton (2008). In Figure 1a, colours mark tasks; in Figure 1b, green and red mark passing and failing inputs. It shows that the eight tasks form well-separated clusters in the input hidden state space. More importantly, even in the reduced dimension, we observe that passing and failing inputs tend to separate well, suggesting that internal hidden states contain information that can distinguish easier inputs from the more challenging ones.

We further examined how failures align with the distribution of passing inputs: we fitted Principal Component Analysis (PCA) Abdi and Williams (2010) on the activations of *passing* inputs from the SPELL-CHK task, and then transformed all inputs from the same SPELL-CHK task, including failing ones, into the fitted space (Figure 2a). We observe a clear pattern where failing inputs deviate from the dense region of passing inputs. However, we note that this distributional deviation is task-specific. Figure 2b presents the same SPELL-CHK inputs, but mapped into the PCA space fitted with the passing inputs of a different task, TOPIC-CLS. In this cross-task projection, the separation between passing and failing inputs become notably weaker compared to the within-task projection. From this observation, we posit that distributional deviation in hidden state space of LLMs can indeed guide test adequacy, but only when analysed in a *task-specific* way. PCA was used here because t-SNE does not transfer across different data samples.

Finally, Figure 1a also shows that some tasks yield multi-modal distributions, with several distinct clusters instead of a single dense region. This means that out-of-distribution detection cannot be simply reduced to measuring distance from the centre of one unimodal distribution. It calls instead for models that can capture multi-modality Kim and Yoo (2021).

## 2.2 Extracting Internal Hidden Vectors from Transformer Architecture

We now turn to the problem of extracting internal states from LLMs. Most contemporary LLMs use the Transformer architecture, which maps each token to an embedding vector and then processes it through a stack of Transformer blocks. Each block contains a self-attention sublayer and a feed-forward sublayer, each wrapped with residual connections. At every layer, the model maintains a residual stream: a continuous vector representation that accumulates contextual information. These vectors are what we refer to as the *hidden states*. Defining a hidden space for measuring test adequacy therefore requires two choices: which input token to extract from, and which layer’s residual vector to use.

The token choice is straightforward. In decoder-only models, the hidden state at the *final token position* encodes the entire prefix, thanks to the autoregressive design. We consider two candidate positions: the last token of the input prompt and the last token of the generated output. The final input token is preferable, as we saw earlier that its hidden states can yield clear separations by task difficulty (Figure 1b), while it also would avoid the computational overhead of generating outputs. For this reason, we focus on the final input token.

Layer choice is less obvious. Prior work shows that middle-to-late layers capture semantic abstractions more effectively, for example, in hallucination detection Kossen et al. (2024); Azaria and Mitchell (2023) and adversarial input identification Zhou et al. (2025). We verify this in our setting by examining hidden states of the final input token across all layers in Llama3.1 8B (32 layers in total), using one natural language task (SPELL-CHK) and one programming task (SYN-BUG). Figure 3 illustrates the results for a subset of the layers, visualised with t-SNE, after first reducing the dimensionality to 10 using PCA for computational efficiency. Early layers show little distinction between passing and failing cases. In middle and later layers, however, separation emerges, with inputs clustering more distinctly by outcome. The effect seems strongest in the latter half of the model’s depth.

Guided by this evidence, we extract hidden states from the layer at *two-thirds of the model’s total depth*, corresponding to the last input token. We denote these as LIHS (Last-token Input Hidden States). For comparison, in our empirical work, hidden states of the last output token are denoted LOHS (Last-token Output Hidden States).

## 3 Approach

CLOTHO formulates the input-level test adequacy as the degree of distributional deviation Kim et al. (2019, 2022). To align the *reference* distribution with the given task, without relying on LLM training data, CLOTHO adaptively models the distribution of inputs the target LLM can successfully handle. This is achieved by fitting a Gaussian Mixture Model (GMM) to hidden state embeddings (LIHS) of passing inputs, expanded through strategic sampling. This section describes details of the approach.

### 3.1 Measuring Test Adequacy as Distributional Deviation

Given a labelled reference set of inputs, CLOTHO estimates probability density of *passing* inputs in the space of LLM hidden states. More concretely, we extract LIHS of passing inputs and apply Gaussian Mixture Model to capture their distribution. The use of GMM is motivated by the fact that the distribution of LIHS may not be unimodal (see Section 2.1).

Algorithm 1 outlines the procedure for iteratively sampling and constructing the reference set, and fitting a GMM on the passing reference inputs until the target reference set size (labelling budget) is reached. The GMM is always trained on the subset of reference inputs labelled as “pass” ( $\mathcal{R}_{\text{pass}}$ , line 5-6). We define a passing input as one that produces correct output in the majority of repeated runs. While alternative definitions are possible, this ensures that the learnt distribution captures inputs that the LLM under test can reliably handle, mitigating randomness.

In the main loop (lines 7-19), CLOTHO expands the reference set  $\mathcal{R}$ . We cast this as an active learning process Cohn et al. (1996), where inputs are iteratively selected for labelling. Each round targets inputs that maximise information gain using two complementary strategies: exploitation, which sharpens the boundaries of the known hidden state space, and exploration, which extends coverage into under-represented regions. To balance these goals, CLOTHO divides the batch into  $B_{\text{exploit}}$  and  $B_{\text{explore}}$  (line 9). Exploitation is guided by the GMM’s component assignment entropy on unlabelled inputs (lines 8 and 10). High entropy signals uncertainty about which component an input belongs to, so labelling the most uncertain inputs helps refine the model’s understanding of the distribution. Exploration, in contrast, is driven by diversity: CLOTHO selects inputs with the greatest minimum Euclidean distance from those already in  $\mathcal{R}$  (line 14), following the greedy max-min strategy of Adaptive Random Testing (ART) Chen et al. (2005). Larger distances ensure broader coverage of the hidden state space. Since no prior knowledge of the latent space is available, CLOTHO weights exploitation and exploration equally, setting  $\alpha = 0.5$ : we may consider adaptive fine-tuning in the future.

**Algorithm 1: CLOTHO**


---

**Input:** a set of unlabelled inputs,  $\mathcal{U}$ , the initial reference set of labelled inputs,  $\mathcal{R}_0$ , target reference set size,  $N$ , batch size,  $B$ , exploration ratio,  $\alpha$

**Output:** an input distribution model  $\mathcal{M}_T$  trained on  $\mathcal{R}_{\text{pass}}$  with  $|\mathcal{R}| = N$

```

1  $\mathcal{R} \leftarrow \mathcal{R}_0$  ;
2  $t \leftarrow 0$  ;// iteration count
3 while  $|\mathcal{R}| < N$  do
4   Adapt  $\mathcal{M}_t$  with best representation dim. and # of mixture components w.r.t.  $\mathcal{R}$  ;
5    $\mathcal{R}_{\text{pass}} \leftarrow \{x \in \mathcal{R} \mid \text{label}(x) = \text{pass}\}$  ;
6   Fit GMM  $\mathcal{M}_t$  on  $\mathcal{R}_{\text{pass}}$  ;
7   for  $x \in \mathcal{U}$  do
8      $s_{\text{exploit}}(x) \leftarrow$  entropy of component responsibilities under  $\mathcal{M}_t$  ;
9      $B_{\text{explore}} \leftarrow \lfloor \alpha B \rfloor$ ,  $B_{\text{exploit}} \leftarrow B - B_{\text{explore}}$  ;
10     $C_{\text{exploit}} \leftarrow$  the  $B_{\text{exploit}}$  inputs with the highest  $s_{\text{exploit}}$  values ;
11     $C_{\text{explore}} \leftarrow \emptyset$  ;
12    t for  $j = 1$  to  $B_{\text{explore}}$  do
13      For each  $x \in \mathcal{U} \setminus (C_{\text{exploit}} \cup C_{\text{explore}})$ ,
14      compute  $d(x) = \min_{y \in \mathcal{R} \cup C_{\text{exploit}} \cup C_{\text{explore}}} \text{Dist}(x, y)$  ;
15      Pick  $x^* = \arg \max_x d(x)$  ;
16       $C_{\text{explore}} \leftarrow C_{\text{explore}} \cup \{x^*\}$  ;
17     $C \leftarrow C_{\text{exploit}} \cup C_{\text{explore}}$  ;
18    Execute and obtain human labels for inputs in  $C$  ;
19     $\mathcal{R} \leftarrow \mathcal{R} \cup C$ ,  $\mathcal{U} \leftarrow \mathcal{U} \setminus C$ ,  $t \leftarrow t + 1$  ;
20 return  $\mathcal{M}_t$ 

```

---

Using Algorithm 1, CLOTHO produces a Gaussian Mixture Model (GMM),  $\mathcal{M}_T$ , that estimates the probability density of hidden states from passing inputs for a given task:

$$\mathcal{L}_{\text{pass}}(x) = p_{\theta}(h(x)), \quad h(x): \text{hidden state representation (LIH)}$$

The density  $p_{\theta}(h(x))$  captures how plausible an input is relative to the reference distribution of LIHS from passing cases: higher values indicate that  $x$  falls within regions commonly associated with correct outputs. From this density, we can compute Likelihood-based Surprise Adequacy (LSA) Kim et al. (2019), defined as the negative log likelihood:

$$\text{LSA}(x) = -\log p_{\theta}(h(x))$$

A higher LSA means the input is more surprising compared to the reference distribution and therefore more likely to trigger a failure. LSA can thus be used either to prioritise human labelling efforts or to identify more challenging inputs for testing.

### 3.2 Adaptive Modelling of Passing Input Distribution via GMM

In Algorithm 1, the model  $\mathcal{M}_t$  is iteratively updated as the reference set expands. To improve model accuracy, we adapt two key parameters during this process: the dimensionality of the LIH representations used by the GMM and the number of mixture components in the model. Both adaptations are performed at line 4 of Algorithm 1.

#### 3.2.1 Dimensionality Selection

Training a GMM with limited data benefits from compact input representations; otherwise, training can become slow and inefficient. Our analysis shows that raw residual hidden state vectors are often too high-dimensional for the GMM to handle effectively, and must be reduced. Moreover, we have noted that the optimal dimensionality varies across tasks. To address this, we apply PCA to project raw hidden states into a lower-dimensional latent space and adaptively determine the ideal number of dimensions. Starting from 10 dimensions, CLOTHO explores two alternatives at each update step: one with 10 more and one with 10 fewer dimensions (lower bound is set to five to avoid collapse). It then performs 3-fold cross-validation on the reference set, based on the rank correlation between the GMM predicted

likelihood of passing and the actual pass rate observed from a set of multiple runs of each input. The dimensionality with the highest mean correlation across folds is chosen for the next iteration, since a stronger correlation indicates a more faithful latent representation.

### 3.2.2 Component Adaptation

A GMM requires the number of mixture components, the Gaussian distributions combined to represent the data, as a hyperparameter. Ideally, this matches the number of task-specific hidden state clusters (see Section 2.1), but that information is usually unavailable in advance. To address this, we adapt the number of components using a perplexity-based heuristic that measures how evenly the GMM distributes probability mass across its components. Given mixture weights  $\mathbf{w} = (w_1, \dots, w_K)$ , for a  $K$  component GMM, we define component perplexity as

$$\text{Perplexity}(\mathbf{w}) = \exp\left(-\sum_{k=1}^K w_k \log w_k\right), \quad \text{where } \mathbf{w} = (w_1, \dots, w_K).$$

CLOTHO starts with five mixture components. If perplexity shows that only a few components dominate, we reduce the count by one. If it indicates that many components are actively contributing, we increase the count by one, up to a maximum of 50.

## 4 Experimental Setup

We present our experimental setup in this section.

### 4.1 Research Questions

Our empirical evaluations are designed to answer the following research questions.

- **RQ1.** How accurately does CLOTHO model the latent space of passing inputs as the reference set expands? If the modelling is accurate, the predicted likelihood of CLOTHO would strongly correlate with the actual passing rates of unseen inputs. We answer RQ1 primarily by comparing the Spearman rank correlation computed for CLOTHO and other baseline approaches that can estimate the likelihood of each input producing the correct output before generation. Further, we also investigate how much the input sampling method of CLOTHO affects the correlation.
- **RQ2.** How effective is CLOTHO at prioritizing nontrivial test cases showing high failure rates for efficient testing? To answer RQ2, we predict pass/fail for inputs based on the predicted likelihood of CLOTHO and report ROC-AUC. We also report failure@ $N$ , i.e., the number of failing inputs out of top  $n$  inputs when sorted according to model score: higher failure@ $N$  would indicate the used score can effectively prioritise failure inducing inputs.
- **RQ3.** How do CLOTHO’s adequacy scores complement existing post-generation uncertainty metrics? We answer RQ3 by comparing Spearman rank correlation of CLOTHO to post-generation measurements including internal state variance, semantic entropy, token probability, and token entropy.
- **RQ4.** Do adequacy scores from CLOTHO transfer from open-weight models to closed-weight proprietary models as valid indicators of input difficulty? We address RQ4 by prioritizing inputs with CLOTHO trained on smaller open-weight LLMs and then evaluating that prioritisation on larger closed-weight models, including GPT-4o, Claude, and Gemini.

### 4.2 Target Tasks and Dataset

Table 1 lists the eight tasks, their prompts and input datasets used in our evaluation. We curate these datasets according to three criteria:

- **Data Volume:** We focus on datasets with sufficiently large number of inputs. First, modelling the input distribution itself requires sufficiently large number of samples, which prevents smaller benchmarks from being used. Second, we need additional inputs to check whether the learnt input distribution generalises to unseen inputs.

- **Clear Ground Truths:** Many general Natural Language Understanding benchmarks, such as question answering, yield open-ended outputs that are hard to validate automatically. We instead focus on tasks with unambiguous ground truth labels.
- **Separation of Template and Inputs:** Since CLOTHO targets specific tasks rather than open-ended dialogue, we use datasets with a clear separation between the task template (instructions) and the input (the variable part, per LLM request). This mirrors realistic use cases already supported by LLM development frameworks LangChain (2024); Sharma et al. (2025).

Table 1: Summary of Tasks, Sources, and Verification Methods Used in Our Experiments

Task ID	# Inputs	Template	Input	Aim
ODD-ADD	6,000	P.E. Guide AI (2025)	Randomly Sampled Numbers	To find sum of only odd numbers
GH-TYPO	10,000	Ours	GitHub Typo Corpus Hagiwara and Mita (2019)	To fix typos extracted from GitHub commits
JSON-FIX	6,563	Ours	Synthetic JSON + Bugs de Jong (2022)	To repair invalid JSON
MODEL-EX	9,810	P.E. Guide AI (2025)	ML-ArXiv CShorten (2022), Synthetic Abstracts	To extract ML model names from paper abstracts
POS-TAG	15,359	PromptPex Sharma et al. (2025); Schnabel and Neville (2024)	UD_English-EWT Silveira et al. (2014)	To detect Part-of-Speech tags
SPELL-CHK	10,000	Ours	WordNet sentences Fellbaum (1998) + Misspellings Mitton et al. (1980)	To fix misspelt words
SYN-BUG	20,518	BICS-Dataset Lee et al. (2024)	Python Syntactic Bug Lee et al. (2024)	To find code lines with syntactic bugs
TOPIC-CLS	7,600	PromptPex Sharma et al. (2025)	AG News Zhang et al. (2015)	To classify topics of news articles

The prompt templates for ODD-ADD, MODEL-EX are taken from Prompt Engineering Guide AI (2025), POS-TAG and TOPIC-CLS from PromptPex Sharma et al. (2025), and SYN-BUG from the BICS benchmark. The templates for GH-TYPO, JSON-FIX, and SPELL-CHK are written by authors to fit the corresponding tasks and datasets. For MODEL-EX, we use paper abstracts from ML-ArXiv dataset CShorten (2022) but also synthesise abstracts that use real ML model names using GPT-4o. All prompt templates, input data, as well as scripts used to synthesise inputs, are available from our replication package.

Table 2: Pass rates ( $p = 0.5$ ) of the studied open-weight LLMs on the eight tasks

	ODD-ADD	GH-TYPO	JSON-FIX	MODEL-EX	POS-TAG	SPELL-CHK	SYN-BUG	TOPIC-CLS
Gemma 2 9B	0.59	0.63	0.79	0.66	0.79	0.88	0.40	0.81
Llama 3.1 8B	0.57	0.48	0.67	0.52	0.74	0.80	0.34	0.77
Mistral 7B	0.08	0.29	0.31	0.49	0.68	0.77	0.28	0.75

### 4.3 Models

To evaluate CLOTHO, we use three widely adopted Open-weight LLMs (OLMs): Llama3.1 8B Grattafiori et al. (2024), Gemma 2 9B Team et al. (2024), and Mistral 7B Jiang et al. (2023). For hidden state extraction from LIHS and LOHS, we select layers at roughly two-thirds of model depth, guided by our preliminary analysis in Section 2.1 and prior work Kossen et al. (2024); Azaria and Mitchell (2023). Table 2 reports the pass rates of these models on our tasks: an input counts as passing if it yields the correct output in a majority of generations, more than five out of ten runs. For RQ4, we additionally include three representative proprietary LLMs: GPT-4o mini<sup>3</sup>, Claude Haiku<sup>4</sup>, and Gemini 2.5 Flash Lite<sup>5</sup>. These serve as targets for testing whether adequacy scores learnt from OLMs transfer effectively to closed-weight ones.

### 4.4 Baselines

We compare CLOTHO against two categories of baseline metrics: pre-generation and post-generation.

#### 4.4.1 Pre-generation Adequacy Metrics

We first consider baselines that, like CLOTHO, estimate test adequacy before any output is generated: *Sequence Log Likelihood* (SLL), *Mahalanobis Distance-based Surprise Adequacy* (MDSA) Kim et al. (2020), and *Base Gaussian Mixture Model* (GMM<sub>b</sub>). SLL is the average log-likelihood of input tokens computed using the LLM under test. It requires no labelled data and serves as a simple, label-free adequacy baseline. MDSA computes the Mahalanobis Distance De Maesschalck et al. (2000) between a new input and the distribution of inputs in the randomly sampled

<sup>3</sup><https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/>

<sup>4</sup><https://www.anthropic.com/clause/haiku>

<sup>5</sup><https://deepmind.google/models/gemini/flash/>

reference set. By assuming a unimodal distribution, it cannot capture the more complex structures observed in input space (see Section 2). Finally,  $\text{GMM}_b$  is an ablated version of CLOTHO that omits both the balanced sampling strategy (Section 3.1) and adaptive modelling (Section 3.2), instead fixing the number of components at five and the latent dimensionality at 10, which is the initial parameters of CLOTHO.

#### 4.4.2 Post-generation Adequacy Metrics

Since CLOTHO essentially predicts the correctness of the output induced by the given input, we can compare CLOTHO to post-generation uncertainty metrics: *Token Probability* ( $\text{TOK}_{prob}$ ) & *Token-Entropy* ( $\text{TOK}_{ent}$ ) Manakul et al. (2023), *Semantic Entropy* ( $\text{SEM}_{ent}$ ) Farquhar et al. (2024), and *Last Output-token Hidden State Variance* ( $\text{LOHS}_{var}$ ).  $\text{TOK}_{prob}$  and  $\text{TOK}_{ent}$  are widely used confidence measures computed from a generated sequence: they can be computed by taking the token level likelihood and entropy at each decoding step of a single output. Following prior work, we derive sentence-level scores by averaging token log-probabilities and entropies over the sequence.  $\text{SEM}_{ent}$  requires multiple outputs from the same prompt, and computes the entropy across semantic clusters formed by the embeddings of outputs. We cluster semantically equivalent outputs among 10 generations using a DeBERTa-based NLI model He et al. (2020), before computing entropy over the resulting clusters. Finally,  $\text{LOHS}_{var}$  captures the variance among Last Output-token Hidden States by taking the trace of the covariance matrix of the collected LOHS vectors. This score reflects how much the model’s internal state diverge when producing different responses to the same prompt, thereby serving as a internal measure of generation variance.

### 4.5 Evaluation Protocols and Metrics

Here we describe the process of experimentation as well as the evaluation metrics used.

#### 4.5.1 Reference Sets and Implementation of CLOTHO

CLOTHO requires a small initial reference set as a starting point. For each task, we provide 10 input-output pairs that are correct as the initial reference set: these are manually crafted and not from the datasets listed in Table 1. This is intended to simulate a realistic use case, in which a developer has a small set of working inputs for a prompt, before starting a more rigorous testing process. One exception is SYN-BUG, for which we find it difficult to craft reliably passing inputs (i.e., inputs with passing rate greater than 0.5), however simple the input is. Consequently, the initial reference set for SYN-BUG contains fewer than ten reliably passing inputs. The difficulty of this task is also noted in the original paper Lee et al. (2024). Note that the same initial reference set is used by CLOTHO, MDSA, and  $\text{GMM}_b$  for RQ1.

CLOTHO is implemented using `GaussianMixture` from `sklearn.mixture` with full covariance matrices Pedregosa et al. (2011). Training terminates when the log-likelihood improvement falls below  $10^{-3}$  or after 500 iterations, whichever comes first. Each iteration uses a batch size of 10, and the maximum target reference set size is 500. To account for randomness in GMM training, the entire process is repeated with ten different random seeds.

#### 4.5.2 Evaluation Metrics

We measure the Spearman rank correlation between the predicted likelihoods and the actual pass rates observed from multiple runs of the same input (we repeat ten times). Spearman rank correlation measures how strongly two sets of ranks are correlated with each other. For RQ2, we report ROC-AUC for binary failure prediction, labelling inputs with pass rate greater than 0.5 as passing. We apply the Mann-Whitney U test to assess statistical significance, and measure failure@ $N$ , the number of failing inputs among the top  $N$  ranked by the score.

## 5 Results

In this section, we present our evaluation results.

### 5.1 Failure Prediction Performance (RQ1)

We begin by evaluating how well CLOTHO can model the distribution of passing inputs in the latent space, and compare CLOTHO to other approaches that are applicable at the pre-generation stage.

#### 5.1.1 RQ1-1: Score Correlation with Actual Pass Rate Observed

Table 3 compares the correlation between actual pass rates from ten inferences and the predicted likelihood of passing obtained from CLOTHO to baselines. Scores derived from distributional deviation - CLOTHO, MDSA, and  $\text{GMM}_b$ -

Table 3: Comparison of pre-generation metrics across reference set sizes (Spearman rank correlation between predicted likelihoods and observed pass rates, higher values indicates better prediction performance).

Task ID	N	Gemma				Llama				Mistral			
		CLOTHO	MDSA	GMM <sub>b</sub>	SLL	CLOTHO	MDSA	GMM <sub>b</sub>	SLL	CLOTHO	MDSA	GMM <sub>b</sub>	SLL
ODD-ADD	100	<b>0.451</b>	0.190	0.082		<b>0.558</b>	0.333	0.087		<b>0.371</b>	0.261	0.215	
	300	<b>0.407</b>	0.212	0.157	-0.201	<b>0.542</b>	0.312	0.207	-0.336	<b>0.389</b>	0.317	0.081	-0.434
	500	<b>0.434</b>	0.184	0.158		<b>0.551</b>	0.239	0.173		<b>0.432</b>	0.356	0.089	
GH-TYPO	100	0.302	0.451	<b>0.472</b>		<b>0.431</b>	0.430	0.392		<b>0.195</b>	0.125	0.126	
	300	0.305	<b>0.468</b>	0.364	-0.046	0.392	<b>0.469</b>	0.439	-0.039	<b>0.189</b>	0.143	0.180	-0.037
	500	0.440	<b>0.455</b>	0.330		0.456	<b>0.482</b>	0.434		<b>0.188</b>	0.129	0.138	
JSON-FIX	100	<b>0.294</b>	0.243	0.200		<b>0.289</b>	0.153	0.072		<b>0.443</b>	0.354	0.253	
	300	<b>0.359</b>	0.238	0.218	-0.005	<b>0.339</b>	0.152	0.238	0.061	<b>0.525</b>	0.401	0.335	0.032
	500	<b>0.377</b>	0.263	0.237		<b>0.377</b>	0.158	0.322		<b>0.562</b>	0.399	0.431	
MODEL-EX	100	<b>0.452</b>	0.223	0.261		<b>0.443</b>	0.358	0.394		<b>0.529</b>	0.404	0.418	
	300	<b>0.398</b>	0.201	0.311	-0.185	<b>0.494</b>	0.392	0.353	-0.430	<b>0.533</b>	0.368	0.362	-0.326
	500	<b>0.379</b>	0.187	0.345		<b>0.505</b>	0.356	0.344		<b>0.550</b>	0.377	0.379	
POS-TAG	100	0.188	0.089	<b>0.196</b>		<b>0.152</b>	0.038	0.094		<b>0.072</b>	-0.026	0.024	
	300	0.248	0.104	<b>0.322</b>	-0.042	<b>0.197</b>	0.010	0.189	0.030	0.116	-0.015	<b>0.139</b>	-0.026
	500	0.252	0.097	<b>0.327</b>		0.203	0.013	<b>0.213</b>		<b>0.143</b>	-0.026	0.143	
SPELL-CHK	100	0.469	0.460	<b>0.475</b>		0.401	0.441	<b>0.483</b>		<b>0.278</b>	0.235	0.235	
	300	0.427	<b>0.453</b>	0.434	-0.103	0.376	0.429	<b>0.433</b>	0.109	0.245	0.249	<b>0.254</b>	0.032
	500	0.441	<b>0.445</b>	0.430		0.384	0.438	<b>0.453</b>		0.237	0.249	<b>0.254</b>	
SYN-BUG	100	0.564	<b>0.564</b>	0.221		<b>0.223</b>	0.134	0.034		<b>0.239</b>	0.239	0.114	
	300	<b>0.627</b>	0.574	0.525	-0.071	<b>0.314</b>	0.110	0.094	-0.050	<b>0.363</b>	0.332	0.296	-0.041
	500	<b>0.696</b>	0.590	0.633		<b>0.328</b>	0.125	0.148		<b>0.391</b>	0.332	0.321	
TOPIC-CLS	100	0.181	0.162	<b>0.208</b>		0.192	0.131	<b>0.206</b>		<b>0.152</b>	0.062	0.096	
	300	0.247	0.139	<b>0.349</b>	0.098	0.224	0.111	<b>0.324</b>	0.142	0.157	0.062	<b>0.216</b>	0.107
	500	0.242	0.133	<b>0.339</b>		0.251	0.114	<b>0.345</b>		0.170	0.062	<b>0.231</b>	
Best Conf. Count		<b>11</b>	5	8	0	<b>15</b>	2	7	0	<b>19</b>	0	5	0

show weak to moderate correlations with observed pass rates, while SLL does not exhibit any meaningful correlation. Among the three distribution-based methods, CLOTHO achieves the highest correlation in 14 of the 24 configurations (8 tasks  $\times$  3 LLMs), while MDSA and GMM<sub>b</sub> perform best in only 3 and 7 configurations, respectively.

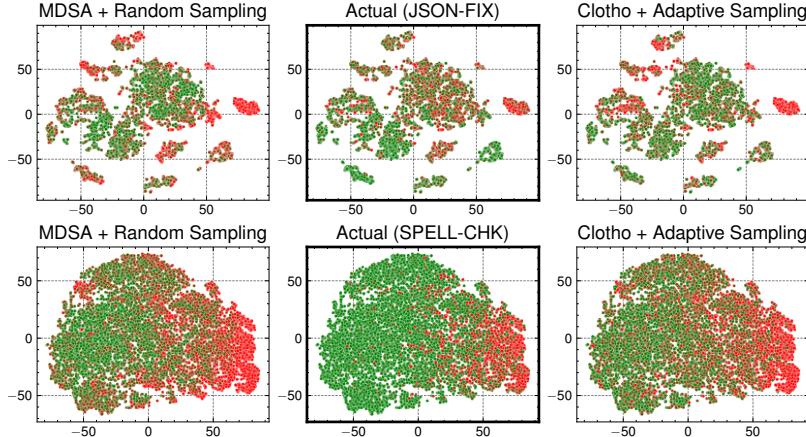


Figure 4: Projection of prediction scores from CLOTHO, MDSA, and GT pass rates (LLaMA,  $N=500$ ).

We observe that CLOTHO outperforms MDSA on some tasks (e.g., JSON-FIX), while the opposite happens for other tasks (e.g., SPELL-CHK). We posit that this can be explained by the structures in input distribution. Figure 4 shows the latent space of LIHS for all inputs of JSON-FIX and SPELL-CHK, each with predicted likelihood of passing from MDSA (left) and CLOTHO (right), as well as the actual ground truth pass rates (centre). We observe that the input distribution of JSON-FIX task has complex structures with multiple clusters, and the unimodal distribution used by MDSA cannot capture the likelihood of passing over the complex distribution. For example, compare the pass and fail prediction for the bottom left cluster of JSON-FIX: CLOTHO predicts most of the inputs to pass (which aligns with the

ground truths), whereas MDSA predicts most of them to be incorrect primarily because they are far from the mean of the main unimodal distribution. In contrast, the input distribution of SPELL-CHK task forms a large single cluster, where the centres of the passing and failing inputs are found in the left and the right part of the cluster, respectively. In cases like this, the distance from the mean of the unimodal distribution may be a reasonably accurate predictor of passing.

Similarly,  $\text{GMM}_b$  outperforms CLOTHO in three tasks: SPELL-CHK, GH-TYPO, and TOPIC-CLS. Note that  $\text{GMM}_b$  has two major differences from CLOTHO: it samples inputs into the reference set randomly, and its GMM does not update component number and representation dimensionality adaptively. We note that GH-TYPO exhibits a unimodal distribution of inputs similar to that of SPELL-CHK. The structure of input distribution for TOPIC-CLS is more complex, but it is more contiguous when compared to tasks like JSON-FIX. Our conjecture is that the smaller number of mixture components used by  $\text{GMM}_b$  (which is the initial value, five), as well as random sampling of inputs into the reference set, may perform reasonably well for tasks like these. Overall, the observed differences support the use of balanced sampling and adaptive modelling used by CLOTHO.

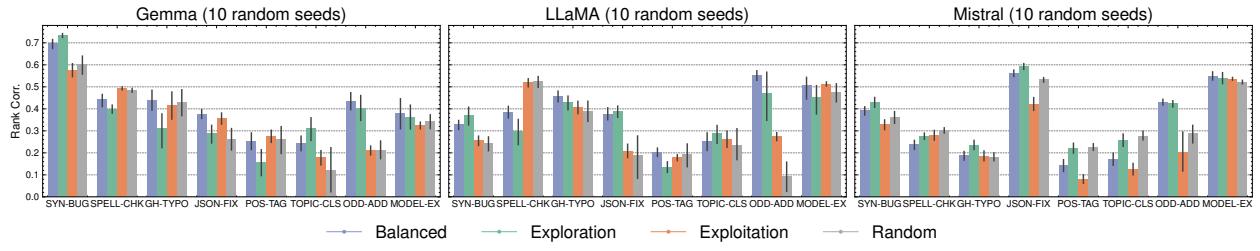


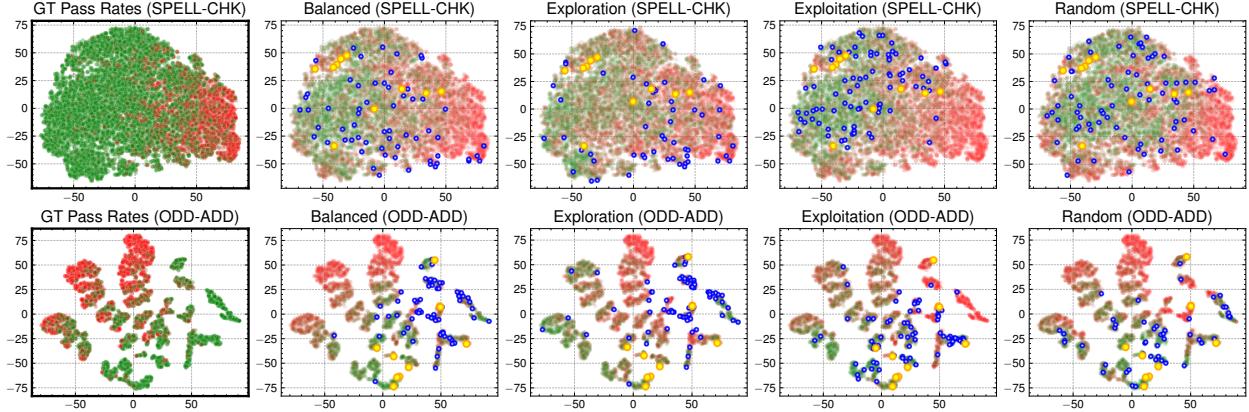
Figure 5: Spearman Rank Correlations from Different Sampling Strategies ( $N=500$ )

### 5.1.2 RQ1-2: Comparison of Different Sampling Strategies

We now consider the impact of sampling strategy in more detail. Figure 5 compares the correlation between predicted likelihood of passing and actual pass rates, obtained with different sampling strategies, when we build a reference set of 500 inputs. CLOTHO’s balanced sampling strategy shows the best average performance, 16.06% higher correlation than random and 2.54% higher than exploration-based sampling (the second best). However, it also shows that, depending on tasks, the best sampling strategy can differ: on Gemma and Llama, balanced sampling outperforms exploration by 10.05% and 8.05%, respectively, while on Mistral, exploration surpasses balanced by 10.17%. We analyse the input distribution in more detail to explain these differences.

Figure 6 shows 100 reference inputs for SPELL-CHK and ODD-ADD, sampled by different sampling strategies from Llama, denoted in blue. The yellow points, on the other hands, are the ten inputs in the initial reference set. Each plot additionally shows the resulting CLOTHO scores for all the remaining inputs in red (likely to fail) and green (likely to pass), which can be compared to the ground truth pass rate distribution shown in GT Pass Rate (leftmost). Inputs for SPELL-CHK form a single contiguous cluster with two polar ends, whereas inputs for ODD-ADD form multiple, fragmented clusters. We note that exploration-based sampling for SPELL-CHK tends to prefer inputs that are outliers, i.e., far from the already sampled reference set. The shape of distribution of SPELL-CHK means that such outliers are more likely to fail, making little contribution to the modelling of the passing input distribution. In contrast, exploitation-based sampling achieves a better coverage of passing inputs, as it tries to clarify the boundaries between mixture components fitted to the initially given passing inputs. Random sampling, on the other hand, achieves a good coverage of the entire input space. Both exploitation-based and random sampling result in more accurate modelling of the input distribution, leading to higher correlation than exploration-based sampling. In contrast, ODD-ADD has fragmented and scattered clusters. Exploration-based sampling successfully discovers outer clusters and accurately models their distributions, whereas exploitation-based and random sampling fail to do so, resulting in lower correlations.

A closer examination of Figure 5 also reveals that Mistral shows a different pattern from those of Llama and Gemma: exploitation-based sampling performs poorly in general, whereas random sampling performs better than for Llama and Gemma. We conjecture that this is because Mistral exhibits substantially higher failure rates across all tasks. For example, in JSON-FIX task, only 31% of inputs achieve pass rates above 0.5, compared to 67% for Llama and 80% for Gemma. Such lower pass rate results in a larger number of smaller, more fragmented clusters of passing inputs. Such fragmented distributions are more difficult for CLOTHO to model, resulting in lower performance for exploitation-based sampling strategy.

Figure 6: Distribution of selected reference points with each sampling strategy (Llama,  $N=100$ )Table 4: ROC-AUC scores and  $p$ -value from Mann-Whitney U Test pass/fail prediction ( $N = 500$ ).

Task ID	ROC-AUC						$p$ -value (Mann-Whitney)					
	Gemma		Llama		Mistral		Gemma		Llama		Mistral	
	CLOTHO	MDSA	CLOTHO	MDSA	CLOTHO	MDSA	CLOTHO	MDSA	CLOTHO	MDSA	CLOTHO	MDSA
ODD-ADD	0.714 ± 0.04	0.609 ± 0.01	0.790 ± 0.02	0.609 ± 0.02	0.883 ± 0.01	0.823 ± 0.03	2.4e-81	4.1e-34	6.1e-254	2.3e-17	6.3e-117	6.5e-84
GH-TYPO	0.737 ± 0.05	0.756 ± 0.02	0.731 ± 0.02	0.754 ± 0.01	0.587 ± 0.02	0.575 ± 0.01	9.3e-105	1.2e-293	5.6e-238	0.0e+00	2.4e-11	2.2e-23
JSON-FIX	0.764 ± 0.02	0.642 ± 0.01	0.690 ± 0.02	0.583 ± 0.01	0.793 ± 0.01	0.702 ± 0.02	4.6e-122	7.0e-45	1.8e-73	8.0e-20	1.4e-253	7.7e-84
MODEL-EX	0.696 ± 0.07	0.595 ± 0.02	0.758 ± 0.05	0.684 ± 0.02	0.788 ± 0.02	0.702 ± 0.02	2.6e-36	2.8e-24	6.3e-104	4.0e-141	0.0e+00	3.0e-183
PUS-TAG	0.648 ± 0.04	0.568 ± 0.01	0.601 ± 0.02	0.516 ± 0.01	0.571 ± 0.02	0.508 ± 0.01	4.4e-41	1.4e-17	2.6e-45	1.8e-02	1.7e-09	2.1e-01
SPELL-CHK	0.869 ± 0.04	0.871 ± 0.01	0.748 ± 0.03	0.776 ± 0.01	0.649 ± 0.02	0.663 ± 0.01	2.6e-186	0.0e+00	2.4e-182	7.0e-266	1.9e-52	1.2e-96
SYN-BUG	0.905 ± 0.02	0.836 ± 0.01	0.714 ± 0.02	0.593 ± 0.01	0.717 ± 0.01	0.679 ± 0.01	0.0e+00	0.0e+00	0.0e+00	1.5e-55	0.0e+00	4.8e-284
TOPIC-CLS	0.640 ± 0.04	0.588 ± 0.01	0.618 ± 0.04	0.557 ± 0.02	0.580 ± 0.03	0.539 ± 0.01	1.9e-29	3.2e-19	4.8e-10	1.7e-06	2.0e-06	3.7e-04
Average	0.746 ± 0.10	0.683 ± 0.11	0.706 ± 0.07	0.634 ± 0.09	0.696 ± 0.11	0.649 ± 0.10	-	-	-	-	-	-

**Answer to RQ1:** CLOTHO successfully predicts pass/fail likelihoods of inputs before the LLM generates outputs, showing clearer advantages over baselines on complex, multi-cluster input distributions. Sampling strategy significantly impacts performance; across tasks, balanced sampling yields the most stable results.

## 5.2 Test Prioritisation Effectiveness (RQ2)

RQ2 concerns how well CLOTHO can prioritise inputs that are likely to fail for more efficient testing. Table 4 shows the ROC-AUC of adequacy scores from CLOTHO and MDSA for the prediction of failure, as well as the  $p$ -values from Mann-Whitney U test of the adequacy scores between passing and failing inputs. CLOTHO achieves ROC-AUC score of 0.716 for unseen inputs across all three models (i.e., those not included in the reference set), when using 500 labelled references (which are, on average, 5.4% of inputs for tasks): in contrast, MDSA achieves ROC-AUC of 0.655. For all tasks and models, both CLOTHO and MDSA produce adequacy scores that can distinguish passing and failing inputs with statistical significance.

Table 5 shows failure@ $N$ , i.e., the number of failure-inducing inputs (i.e., those whose pass rates are 0.5 or below) out of  $N$  inputs selected in the descending order of adequacy metrics ( $N = \{100, 300, 500\}$ ): for each pair of metric and  $N$ , we report average and standard deviation of failure@ $N$  from 10 different orderings, each obtained from repeated runs. Note that the column Random represents random ordering and, therefore, the corresponding failure@ $N$  values reflect the average ratio of failure-inducing inputs in the entire dataset for each task. The best failure@ $N$  for each configuration is typeset in bold. CLOTHO shows strong capability in prioritizing failure-inducing inputs, with 80.4% of the top-100 ranked inputs and 75.2% of top-300 ranked inputs exposing actual failures, compared to 73.1% and 68.2% for MDSA, respectively, when aggregated over all tasks. Notably, CLOTHO can significantly outperform MDSA on tasks where MDSA struggles (e.g., ODD-ADD or MODEL-EX). When MDSA outperforms CLOTHO, the margins tend to be smaller. These results highlight the practical value of CLOTHO: by ranking inputs, it helps testers focus execution and labelling effort on the most challenging cases.

**Answer to RQ2:** CLOTHO’s learnt adequacy scores can reliably identify failure-inducing inputs on unseen data. For test prioritisation, CLOTHO consistently ranks a high proportion of failures in the top positions, providing evidence that it can help testers focus labelling effort on the most challenging inputs.

Table 5: Comparison of failure@N for all tasks and models ( $\mu$ : mean,  $\sigma$ : standard deviation).

Task ID	N	Gemma						Llama						Mistral														
		CLOTHO			MDSA			Random			CLOTHO			MDSA			Random			CLOTHO			MDSA			Random		
		$\mu$	$\sigma$	$\mu$	$\sigma$	$\mu$	$\sigma$	$\mu$	$\sigma$	$\mu$	$\sigma$	$\mu$	$\sigma$	$\mu$	$\sigma$	$\mu$	$\sigma$	$\mu$	$\sigma$	$\mu$	$\sigma$	$\mu$	$\sigma$	$\mu$	$\sigma$	$\mu$	$\sigma$	
ODD-ADD	100	<b>100.0</b>	0.00	<b>100.0</b>	0.00	40.7	4.06	<b>95.9</b>	7.22	29.0	18.62	38.1	3.11	<b>99.9</b>	0.32	96.4	4.43	91.4	1.90									
	300	299.5	0.71	<b>300.0</b>	0.00	124.7	9.08	<b>284.0</b>	11.98	123.9	59.64	117.2	5.53	<b>299.7</b>	0.95	293.5	6.36	272.3	4.79									
	500	455.4	13.31	<b>477.3</b>	6.72	200.8	9.27	<b>459.9</b>	14.39	219.1	77.32	198.8	10.59	<b>499.4</b>	1.58	492.7	7.06	455.4	6.22									
GH-TYPO	100	<b>86.7</b>	5.01	77.5	7.75	34.6	3.86	<b>95.7</b>	2.06	82.4	2.80	48.4	4.14	<b>92.0</b>	3.20	83.7	2.79	69.5	4.79									
	300	<b>247.6</b>	17.15	246.3	15.37	106.8	5.18	<b>278.0</b>	11.10	251.4	10.34	146.2	9.45	<b>259.3</b>	18.54	247.9	5.38	206.1	12.84									
	500	402.2	33.27	<b>408.3</b>	23.98	179.9	6.06	<b>453.2</b>	18.28	249.7	13.54	244.3	13.43	<b>417.4</b>	26.53	404.0	11.12	342.0	12.63									
JSON-FIX	100	<b>76.7</b>	7.09	67.5	9.03	16.2	3.77	<b>92.3</b>	4.00	89.8	0.79	24.9	5.30	<b>99.9</b>	0.32	<b>100.0</b>	0.00	58.8	7.02									
	300	<b>220.7</b>	24.33	133.9	7.99	51.4	5.76	230.3	8.62	<b>234.3</b>	8.17	75.6	6.40	<b>296.9</b>	1.20	295.6	0.52	177.0	11.82									
	500	326.3	39.50	195.0	13.68	85.9	4.07	<b>316.7</b>	10.14	305.6	11.04	125.5	10.56	<b>478.3</b>	4.52	454.7	16.50	299.0	10.67									
MODEL-EX	100	<b>45.3</b>	17.64	27.0	2.45	33.9	3.84	<b>69.7</b>	6.04	39.0	7.15	44.0	4.71	<b>75.7</b>	5.50	41.0	5.85	46.5	3.69									
	300	<b>143.2</b>	41.57	88.0	7.21	101.6	7.18	<b>205.2</b>	16.39	132.1	8.05	134.2	9.92	<b>231.3</b>	10.65	146.4	13.55	144.6	9.36									
	500	<b>240.0</b>	65.77	162.9	13.80	167.1	10.65	<b>337.5</b>	19.85	245.7	10.73	224.7	13.86	<b>383.5</b>	12.96	278.1	17.93	245.5	12.31									
POS-TAG	100	76.0	13.60	<b>78.2</b>	1.75	22.3	3.80	64.3	7.93	<b>74.6</b>	1.26	25.1	4.77	86.6	4.90	<b>91.5</b>	0.53	28.0	3.43									
	300	<b>199.6</b>	39.97	190.2	12.17	59.1	4.12	154.0	13.17	<b>166.7</b>	9.38	69.3	7.41	218.4	12.96	<b>224.0</b>	2.00	87.2	7.91									
	500	282.6	53.80	<b>283.4</b>	19.02	99.7	5.76	<b>239.3</b>	24.53	228.4	8.66	116.7	9.82	<b>325.6</b>	21.88	320.4	8.75	148.7	8.84									
SPELL-CHK	100	<b>86.9</b>	2.60	82.0	0.82	12.3	4.32	<b>75.3</b>	6.17	69.2	1.55	18.3	3.68	<b>70.9</b>	5.04	60.1	3.00	21.7	3.20									
	300	<b>238.9</b>	6.08	223.0	12.25	37.5	7.14	<b>208.8</b>	19.01	207.7	10.49	56.3	5.29	<b>199.5</b>	16.03	170.8	15.63	61.6	6.59									
	500	<b>362.2</b>	14.73	331.8	19.56	61.8	7.04	313.4	31.92	<b>335.2</b>	23.08	92.9	11.10	<b>301.4</b>	19.78	264.8	19.53	104.5	6.60									
SYN-BUG	100	94.4	3.17	<b>96.2</b>	2.86	58.9	4.68	<b>84.8</b>	5.49	76.7	3.56	60.6	4.67	87.4	4.95	<b>88.1</b>	4.75	69.4	4.62									
	300	285.1	6.67	<b>287.8</b>	7.84	173.0	8.25	<b>254.1</b>	11.24	231.7	7.73	188.6	8.71	<b>267.5</b>	8.06	261.8	12.66	206.3	7.39									
	500	477.7	8.60	<b>480.5</b>	9.78	286.6	8.66	<b>420.1</b>	15.07	383.6	14.84	312.0	9.10	<b>448.9</b>	11.01	438.9	16.31	341.0	7.90									
TOPIC-CLS	100	52.0	10.41	<b>59.9</b>	3.51	15.5	3.31	52.0	9.73	<b>64.9</b>	3.35	20.2	3.19	49.8	6.66	<b>62.2</b>	2.04	25.7	3.71									
	300	129.3	22.70	<b>154.0</b>	8.38	50.8	5.73	123.1	16.06	<b>154.7</b>	7.21	62.6	5.74	121.6	14.51	<b>129.4</b>	5.10	70.4	8.10									
	500	193.6	29.28	<b>217.2</b>	15.52	88.4	11.21	185.6	26.62	<b>223.6</b>	9.61	107.2	7.15	<b>186.1</b>	15.32	177.4	6.11	115.6	8.80									

### 5.3 RQ3: Complementarity Analysis

RQ3 compares CLOTHO to post-generation approaches. Table 6 begins by comparing the results of correlation analysis, i.e., how well the adequacy scores correlate with pass rates, across CLOTHO and post-generation metrics: the highest are typeset in bold, and the second highest underlined. Post-generation variance based metrics generally perform well, with  $\text{LOHS}_{var}$  achieving the strongest correlations overall (6/8 for Llama, 5/8 for Mistral). In contrast, token-based confidence measures generally show only moderate correlations across all configurations. Overall, CLOTHO demonstrates strong performance, despite lacking access to the actual model outputs. Notably, CLOTHO achieves the best performance on specific tasks, particularly for SYN-BUG, where post-generation approaches typically underperform. We also note that the performance distribution varies significantly across model architectures. For Gemma, CLOTHO shows competitive performance compared to variance-based measures, suggesting particular compatibility with certain model families.

Given the variance in correlations, we investigate whether these metrics can complement each other in order to detect inputs with high failure rates. First, we choose inputs that fail eight or more times out of 10 repeated runs. Subsequently, we compute three representative adequacy metrics: CLOTHO,  $\text{LOHS}_{var}$ ,  $\text{TOK}_{ent}$ , and choose the inputs with the top 25% scores. Figure 7 shows Venn diagram of such high adequacy, high failure rate inputs for four tasks: SYN-BUG, GH-TYPO, ODD-ADD, and POS-TAG. The number of covered inputs shows how many of the high failure rate inputs are also considered as high adequacy inputs by the three metrics analysed: collectively, at least 55.7% (SYN-BUG) or more of those inputs show top 25% adequacy scores by at least one of the metrics. The diagrams also show the complementary relationship between the metrics: unanimous agreements between three metrics range only from 5.7% to 15% of high failure rate inputs, while individual metrics uniquely account for up to 22.3% of them (CLOTHO for ODD-ADD). This suggests that combining the metrics can broaden coverage and capture a larger portion of failure-inducing inputs, indicating that although CLOTHO was originally designed as a pre-generation metric, it can also provide complementary benefits when integrated with post-generation metrics to better focus labelling efforts.

Table 6: Comparison between CLOTHO and post-generation uncertainty metrics across all models and tasks.

Task ID	Gemma				Llama				Mistral							
	$\text{LOHS}_{var}$	$\text{SEM}_{ent}$	$\text{TOK}_{prob}$	$\text{TOK}_{ent}$	CLOTHO	$\text{LOHS}_{var}$	$\text{SEM}_{ent}$	$\text{TOK}_{prob}$	$\text{TOK}_{ent}$	CLOTHO	$\text{LOHS}_{var}$	$\text{SEM}_{ent}$	$\text{TOK}_{prob}$	$\text{TOK}_{ent}$	CLOTHO	
ODD-ADD	0.390	<b>0.487</b>	-0.138	<u>0.458</u>	0.434	0.519	<b>0.730</b>	0.548	0.467	<u>0.551</u>	-0.076	<b>0.282</b>	0.004	0.005	<b>0.432</b>	
GH-TYPO	0.534	<u>0.558</u>	0.284	<u>0.540</u>	0.440	<b>0.800</b>	<u>0.543</u>	0.382	0.465	<u>0.456</u>	<u>0.614</u>	<b>0.633</b>	0.325	0.430	0.188	
JSON-FIX	<u>0.342</u>	0.308	0.138	0.230	<u>0.377</u>	<b>0.688</b>	<u>0.442</u>	0.277	0.319	0.377	<u>0.632</u>	0.403	0.226	0.246	<u>0.562</u>	
MODEL-EX	<u>0.348</u>	0.124	-0.184	0.177	<u>0.379</u>	<b>0.650</b>	0.223	0.173	0.196	<u>0.505</u>	<u>0.581</u>	0.237	0.062	0.051	<u>0.550</u>	
POS-TAG	<u>0.310</u>	<b>0.313</b>	0.026	0.218	0.252	<u>0.726</u>	<u>0.367</u>	0.274	0.321	0.203	<u>0.693</u>	<u>0.410</u>	0.221	0.263	0.143	
SPELL-CHK	<u>0.461</u>	0.425	0.263	0.415	<u>0.441</u>	<b>0.722</b>	<u>0.526</u>	0.430	0.480	0.384	<u>0.520</u>	<u>0.495</u>	0.233	0.279	0.237	
SYN-BUG	-0.069	0.090	0.007	0.055	<b>0.696</b>	0.161	0.194	0.188	<u>0.219</u>	<u>0.328</u>	-0.168	-0.092	-0.040	-0.036	<u>0.391</u>	
TOPIC-CLS	<u>0.364</u>	0.174	0.039	0.121	<u>0.242</u>	<u>0.556</u>	0.175	0.142	0.152	<u>0.251</u>	<u>0.597</u>	0.234	0.139	0.180	0.170	

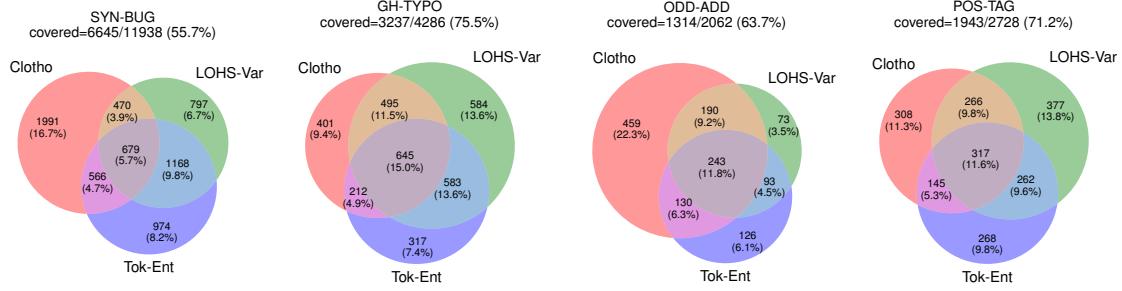


Figure 7: 3-way Venn diagrams drawn from samples with high adequacy scores (top 25%) each from CLOTHO, LOHS<sub>var</sub>, and TOK<sub>ent</sub> among high-failure-rate inputs ( $p_f \geq 0.8$ ) in Llama.

**Answer to RQ3:** CLOTHO’s failure prediction performance is comparable to those of post-generation metrics, despite not having access to the actual generated outputs. Closer analysis shows they are complementary, each capturing distinct subsets of failure-inducing inputs, indicating the potential of integrating CLOTHO with post-generation metrics.

Table 7: Spearman’s rank correlation between scores obtained from OLMs with reference set size 500 and actual pass rates of proprietary LLMs.

Task ID	Gemma				LLaMA				Mistral			
	SELF	Claude	GPT	Gemini	SELF	Claude	GPT	Gemini	SELF	Claude	GPT	Gemini
ODD-ADD	0.444	<b>0.447</b>	0.094	0.064	0.549	0.411	<b>0.439</b>	0.235	0.434	0.231	<b>0.437</b>	0.182
GH-TYPO	0.438	0.383	0.373	<b>0.442</b>	0.470	0.380	0.365	<b>0.413</b>	0.197	0.106	<b>0.140</b>	0.125
JSON-FIX	0.394	<b>0.128</b>	0.108	0.111	0.360	0.098	<b>0.167</b>	0.160	0.568	<b>0.203</b>	-0.010	0.006
MODEL-EX	0.386	0.248	0.347	<b>0.402</b>	0.512	0.248	0.341	<b>0.427</b>	0.555	0.288	0.380	<b>0.456</b>
POS-TAG	0.289	0.267	<b>0.272</b>	0.253	0.221	0.210	<b>0.235</b>	0.184	0.138	0.190	<b>0.233</b>	0.173
SPELL-CHK	0.452	<b>0.465</b>	0.443	0.419	0.374	<b>0.349</b>	0.340	0.321	0.247	<b>0.292</b>	0.263	0.239
SYN-BUG	0.707	0.435	<b>0.543</b>	0.532	0.345	0.190	0.178	<b>0.224</b>	0.396	0.325	<b>0.379</b>	0.349
TOPIC-CLS	0.252	0.215	<b>0.259</b>	0.229	0.256	0.188	0.201	<b>0.209</b>	0.181	0.139	0.160	<b>0.161</b>

#### 5.4 RQ4: Transferability to Proprietary LLMs

The final research question focuses on the more practical applicability of CLOTHO: by design, CLOTHO requires internal representations of the target LLMs that are only available from Open-weight Language Models (OLMs). However, some application may have to rely on stronger, proprietary closed-weight models. Table 7 reports score correlation analysis results, in which the adequacy scores are computed by CLOTHO using open-weight models while the pass rates are obtained from 2,000 queries made to each of Claude, GPT, and Gemini. The results show that the adequacy scores predicted using OLMs can still produce moderate correlations with the actual pass rates of proprietary models: in some cases, even outperform the correlation against pass rates from the OLM itself (contained in the SELF column). For example, Gemma scores achieve higher correlation when applied to Gemini for MODEL-EX than when applied to itself; similarly, Mistral scores perform better when applied to GPT-4o mini then itself for POS-TAG task. This suggests that CLOTHO can capture certain generalisable aspects of input difficulty that transcend specific model architectures.

We subsequently evaluate the effectiveness of these transferred adequacy scores for prioritizing failure-inducing inputs for the proprietary models. Figure 8 shows the cumulative failure rates for the transferred input orderings for GH-TYPO, JSON-FIX, SPELL-CHK, SYN-BUG, and ODD-ADD for a subset of transfer pairs (best transfer pair for each task): the solid lines denote the average from 10 runs of CLOTHO, and the coloured bands denote the standard deviation. We also report the Average Percentage of Fault Detected (APFD) Elbaum et al. (2000), which is essentially the area under curve. Given that average APFD would converge to 0.5 as the number of runs increases, the transferred ordering can achieve significantly effective test prioritisation. This is because proprietary models tend to have lower failure rate. As such, prioritising failure-inducing inputs early on can have bigger impact in proprietary models than in OLMs. Table 8 contains the failure@ $N$  from transferred CLOTHO scores at  $N=500$ , as well as random orderings, per task: the large gains over random ordering aligns with high APFDs we observe in Figure 8. Aggregated across tasks and models, CLOTHO finds on average 23.75 additional failures at  $N=100$  and 61.15 at  $N=500$  over random selection, corresponding to 126.8% and 64.5% improvements, respectively.

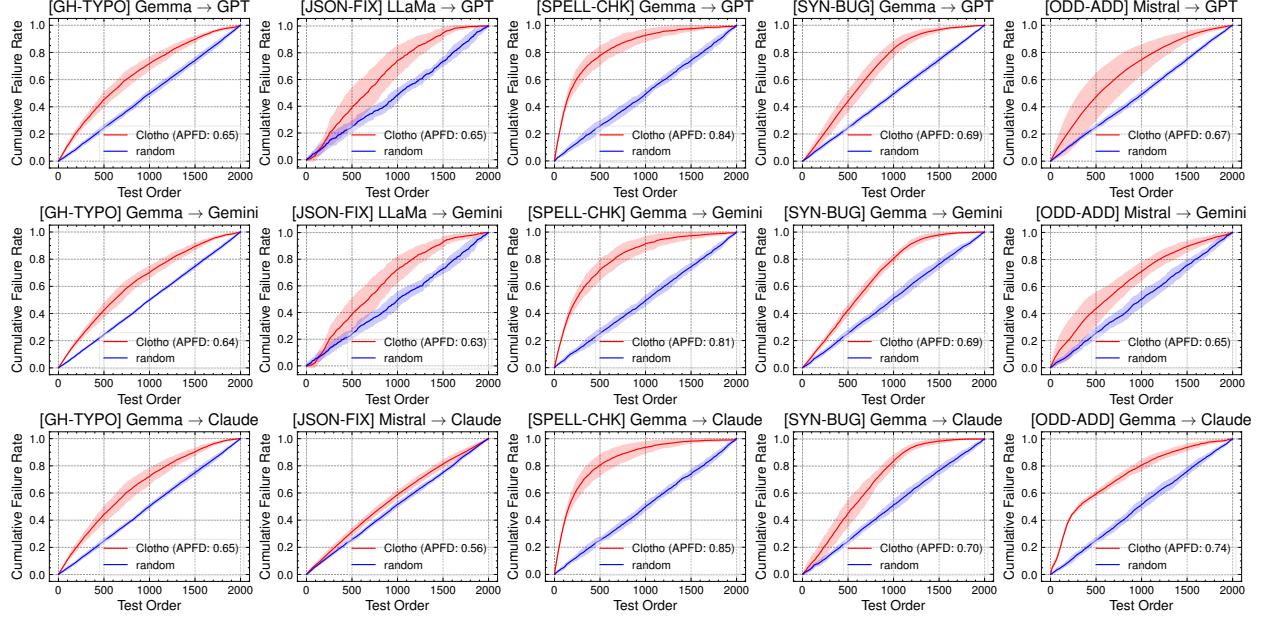


Figure 8: Average percentage of detected failures when scores from Open-weight Language Models (OLMs) are applied to proprietary LLMs.

Table 8: Number of detected failing inputs at  $N$  labelling budgets (failure@ $N$ ) when scores from OLMs are applied to proprietary model ( $N=500$ ).

Task ID	Claude				GPT				Gemini									
	Gemma		Llama		Mistral		Gemma		Llama		Mistral		Gemini					
	CLOTHO	Rand.	CLOTHO	Rand.	CLOTHO	Rand.	CLOTHO	Rand.	CLOTHO	Rand.	CLOTHO	Rand.	CLOTHO	Rand.				
ODD-ADD	165.5	69.9	123.7	69.9	79.8	69.9	103.4	103.3	242.1	103.3	241.5	103.3	42.1	37.8	78.0	37.8	72.2	37.8
GH-TYPO	280.3	156.2	284.0	156.2	201.0	156.2	260.3	140.4	246.3	140.4	184.5	140.4	329.6	189.9	328.3	189.9	232.5	189.9
JSON-FIX	159.9	135.0	172.1	135.0	164.9	135.0	21.3	14.9	26.4	14.9	7.9	14.9	23.6	15.9	28.2	15.9	9.8	15.9
MODEL-EX	137.5	99.7	132.6	99.7	149.6	99.7	220.4	140.7	199.9	140.7	225.3	140.7	298.6	202.6	289.6	202.6	315.2	202.6
POS-TAG	126.3	66.4	108.6	66.4	111.9	66.4	145.0	81.7	137.8	81.7	137.6	81.7	114.4	63.4	99.7	63.4	96.2	63.4
SPELL-CHK	187.4	58.5	152.2	58.5	136.0	58.5	153.2	50.5	128.9	50.5	110.8	50.5	173.1	60.3	137.0	60.3	119.1	60.3
SYN-BUG	111.1	60.5	81.5	60.5	112.0	60.5	321.2	179.0	201.7	179.0	287.2	179.0	143.5	85.1	114.4	85.1	136.0	85.1
TOPIC-CLS	139.4	79.3	122.6	79.3	110.2	79.3	163.4	86.9	133.8	86.9	128.9	86.9	170.0	94.9	149.3	94.9	141.6	94.9

**Answer to RQ4:** Adequacy scores that CLOTHO learns from open-weight models can be successfully transferred to closed-weight LLMs, enabling effective test prioritisation of failure-inducing inputs without any access to the internals of proprietary LLMs.

## 6 Discussion

We discuss implications of our findings and outline directions for future work.

### 6.1 Distinguishing Different Failure Modes

Our evaluation focuses on CLOTHO’s ability to identify inputs that trigger failures, aligning with the core aim of robustness testing: prioritizing failure-inducing cases. Since failures are relatively common in LLMs (as confirmed in our studied tasks), testers may also want to distinguish among *types* of failures. For example, in syntactic bug detection, it is often more valuable to gather one instance from several bug categories (e.g., missing brackets, incorrect keywords) than to collect many instances of the same type.

Since CLOTHO partitions the input space with a Gaussian Mixture Model (GMM), its components may reveal distinct modes of failure. Targeting low-likelihood inputs from different components could increase the diversity of failures found. Beyond this, building models of failing inputs, potentially one per failure category, alongside the passing-input model offers another path to sharpen failure discovery and characterisation.

Another avenue is to distinguish adversarial inputs—*deliberately* crafted to elicit unexpected behaviour—from non-adversarial yet failure-inducing ones. While our evaluation focuses on non-adversarial, task-valid cases, early evidence shows adversarial inputs produce sharply divergent representations with much lower likelihoods, suggesting CLOTHO could help separate the two.

## 6.2 Tasks Beyond Binary Correctness

Tasks in our study, like many existing LLM benchmarks, adopt binary correctness criteria. Yet real-world applications often demand richer evaluation, with graded or multi-dimensional measures such as politeness, factual accuracy, or consistency in a counselling chatbot. We account for non-determinism through majority-pass criteria over repeated runs, but the idea of a “passing” input can naturally extend to thresholding continuous scores or aggregating grades across dimensions, as seen in recent evaluation frameworks PromptFoo (2024); AI (2024a,b); LangChain (2024). We hypothesise that CLOTHO can adapt to such settings by building reference sets around properties of interest and by defining adequacy relative to those properties. Exploring how thresholding and multi-criteria aggregation shape CLOTHO’s performance will be essential to broadening its scope.

Modern LLM use also involves complex interaction patterns, such as tool use in autonomous agents, where correctness unfolds across multiple reasoning steps and each step has distinct failure modes. For instance, a tool-use agent may fail by (1) invoking a tool with invalid arguments, (2) misinterpreting tool outputs, or (3) reasoning incorrectly after the tool call. Applying CLOTHO step by step, estimating adequacy from internal representations before each action, could help predict failures early and guide targeted interventions. An open question is whether CLOTHO can scale to full agent trajectories spanning multiple conversation turns, and how the task-specific internal space shifts over the course of execution.

## 7 Threats to Validity

Internal validity could be threatened because LLMs are not fully interpretable, meaning that CLOTHO might capture latent patterns that correlate with but are not identical to task-specific difficulty; using open-weight models may improve transparency and reproducibility but does not resolve this gap. External validity could also be at risk since CLOTHO learns task- and model-specific input distributions, which might limit generalisation, especially if other architectures such as reasoning LLMs (e.g., DeepSeek R1 DeepSeek-AI et al. (2025)) produce different latent spaces; we attempt to reduce this risk by evaluating multiple models and tasks, though broader studies will be useful. Construct validity could likewise be threatened if our chosen metrics do not fully reflect adequacy, and while we use established measures (ROC-AUC, APFD) and a scrutinised, open-source GMM implementation Pedregosa et al. (2011), these proxies may not perfectly represent the underlying concept.

## 8 Related Work

**Post-Generation Uncertainty:** LLM outputs have been widely studied to measure uncertainty and, consequently, detect hallucinations. Chen et al. propose EigenScore, which leverages the eigenvalue decomposition of activation covariance matrices to detect hallucinations Chen et al. (2024). The MIND approach of Su et al. goes further and toward online use, showing that unsupervised detectors over streaming internal states can flag hallucinations in real time during decoding Su et al. (2024). Factoscope from He et al. aggregates multi-layer inner states with a Siamese network architecture to separate factual from non-factual outputs across models, reinforcing that hidden-state geometry correlates with veracity He et al. (2024). CLOTHO, on the other hand, uses only the internal hidden space of inputs.

**Input-side Internal Activation:** Recent work show that an LLM’s input-side internal activations already contain strong signals about errors in the output. Azaria and Mitchell train a lightweight classifier on hidden states to distinguish true statements from the false, outperforming probability-only baselines and demonstrating that pre-generation representations encode truthfulness cues Azaria and Mitchell (2023). Snyder et al. probe tokens and activations before the hallucination begins and train binary detectors that can forecast hallucinations in factual QA Snyder et al. (2024). Extending this, Kossen et al. introduce Semantic Entropy Probes (SEPs) that approximate semantic entropy from a single forward pass (vs. multi-sample SE Farquhar et al. (2024)), achieving strong detection and improving out-of-distribution generalisation Kossen et al. (2024). CLOTHO also uses the input-side activations, but measures task-specific test adequacy via reference set construction.

**Out-of-distribution Detection:** For deep neural networks, proposed hidden-state OOD detectors assess how atypical an input is relative to training data. Distance-based methods model class-conditional Gaussians and flag inputs with large Mahalanobis distance Lee et al. (2018). Density-based methods fit mixture models over utterance embeddings,

such as GMMs, for unknown intent detection Fan et al. (2020). Reconstruction-based methods rely on hierarchical VAEs Havtorn et al. (2021). Unlike these approaches, which centre on detecting outliers or adversarial cases with respect to training distributions, CLOTHO addresses task-specific test adequacy by adaptively building a labelled reference set and modelling pre-generation hidden states in LLMs.

## 9 Conclusion

We propose CLOTHO, a technique that measures task-specific test adequacy for LLM inputs at the pre-generation stage. It iteratively constructs a set of reference inputs, whose internal latent space captures the distribution of inputs that are likely to pass. CLOTHO uses modelling and an adaptive sampling strategy to reduce the number of reference set inputs that need to be labelled by humans. The learnt distribution is subsequently used to compute task-specific Surprise Adequacy scores that can rank unseen test inputs.

Our empirical evaluation, across diverse tasks, show that CLOTHO accurately models the latent distributions of inputs that are likely to pass, and can achieve effective test prioritisation performance. Importantly, adequacy scores learnt on open-weight LLMs transfer effectively to proprietary models, allowing for cost-efficient testing without repeated and expensive LLM inferences.

Future work include evaluation against a wider range of both open and proprietary models, as well as investigating alternative sampling strategies that can further reduce labelling efforts. More broadly, we see pre-generation adequacy metrics as a promising foundation for scalable, task-aware testing of LLM-based systems, as well as hallucination and adversarial input detection.

## Data Availability

The data for the studied tasks, as well as the code for replication of the findings of this paper, are publicly available from <https://anonymous.4open.science/r/clotho-artifact-DE31>.

## References

Hervé Abdi and Lynne J Williams. 2010. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics* 2, 4 (2010), 433–459.

Confident AI. 2024a. DeepEval. <https://github.com/confident-ai/deepeval>.

DAIR AI. 2025. Prompt Engineering Guide: <https://www.promptingguide.ai>.

Giskard AI. 2024b. Giskard. <https://github.com/Giskard-AI/giskard>.

Amos Azaria and Tom Mitchell. 2023. The Internal State of an LLM Knows When It’s Lying. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore, 967–976. <https://doi.org/10.18653/v1/2023.findings-emnlp.68>

Chao Chen, Kai Liu, Ze Chen, Yi Gu, Yue Wu, et al. 2024. INSIDE: LLMs’ internal states retain the power of hallucination detection. *arXiv preprint arXiv:2402.03744* (2024).

T. Y. Chen, H. Leung, and I. K. Mak. 2005. Adaptive Random Testing. In *Advances in Computer Science - ASIAN 2004. Higher-Level Decision Making*, Michael J. Maher (Ed.). Springer, Berlin, Heidelberg, 320–329. [https://doi.org/10.1007/978-3-540-30502-6\\_23](https://doi.org/10.1007/978-3-540-30502-6_23)

David A Cohn, Zoubin Ghahramani, and Michael I Jordan. 1996. Active learning with statistical models. *Journal of artificial intelligence research* 4 (1996), 129–145.

CShorten. 2022. arXiv Dataset: Hugging Face Dataset Card. <https://huggingface.co/datasets/CShorten/Last-Week-on-ML-ArXiv>. Accessed: 2025-09-11.

Jos de Jong. 2022. JSON Repair: How to fix JSON and validate it with ease. <https://jsoneditoronline.org/indepth/parse/fix-json/>. Accessed: 2025-09-11.

Roy De Maesschalck, Delphine Jouan-Rimbaud, and Désiré L Massart. 2000. The mahalanobis distance. *Chemometrics and intelligent laboratory systems* 50, 1 (2000), 1–18.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, et al. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *arXiv:cs.CL/2501.12948* <https://arxiv.org/abs/2501.12948>

Sebastian G. Elbaum, Alexey G. Malishevsky, and Gregg Rothermel. 2000. Prioritizing test cases for regression testing. In *Proceedings of International Symposium on Software Testing and Analysis*. 102–112.

Lu Fan, Guangfeng Yan, Qimai Li, Han Liu, Xiaotong Zhang, et al. 2020. Unknown Intent Detection Using Gaussian Mixture Model with an Application to Zero-shot Intent Classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (Eds.). Association for Computational Linguistics, Online, 1050–1060. <https://doi.org/10.18653/v1/2020.acl-main.99>

Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. 2024. Detecting hallucinations in large language models using semantic entropy. *Nature* 630, 8017 (2024), 625–630.

Christiane Fellbaum. 1998. *WordNet: An electronic lexical database*. MIT press.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, et al. 2020. The Pile: An 800GB Dataset of Diverse Text for Language Modeling. arXiv:cs.CL/2101.00027 <https://arxiv.org/abs/2101.00027>

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, et al. 2024. The Llama 3 Herd of Models. arXiv:cs.AI/2407.21783 <https://arxiv.org/abs/2407.21783>

Masato Hagiwara and Masato Mita. 2019. GitHub typo corpus: A large-scale multilingual dataset of misspellings and grammatical errors. *arXiv preprint arXiv:1911.12893* (2019).

Jakob D. Høftorn, Jes Frellsen, Søren Hauberg, and Lars Maaløe. 2021. Hierarchical VAEs Know What They Don't Know. In *Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Marina Meila and Tong Zhang (Eds.), Vol. 139. PMLR, 4117–4128. <https://proceedings.mlr.press/v139/høftorn21a.html>

Jinwen He, Yujia Gong, Zijin Lin, Cheng'an Wei, Yue Zhao, et al. 2024. LLM Factoscope: Uncovering LLMs' Factual Discernment through Measuring Inner States. In *Findings of the Association for Computational Linguistics: ACL 2024*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, Bangkok, Thailand, 10218–10230. <https://doi.org/10.18653/v1/2024.findings-acl.608>

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654* (2020).

Yuheng Huang, Jiayang Song, Zhijie Wang, Shengming Zhao, Huaming Chen, et al. 2025. Look before you leap: An exploratory study of uncertainty analysis for large language models. *IEEE Transactions on Software Engineering* (2025).

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, et al. 2023. Mistral 7B. arXiv:cs.CL/2310.06825 <https://arxiv.org/abs/2310.06825>

Jinhan Kim, Robert Feldt, and Shin Yoo. 2019. Guiding Deep Learning System Testing using Surprise Adequacy. In *Proceedings of the 41th International Conference on Software Engineering (ICSE 2019)*. IEEE Press, 1039–1049.

Jinhan Kim, Robert Feldt, and Shin Yoo. 2022. Evaluating Surprise Adequacy for Deep Learning System Testing. *ACM Transactions on Software Engineering and Methodology* 32, 2 (June 2022), 1–29.

Jinhan Kim, Jeongil Ju, Robert Feldt, and Shin Yoo. 2020. Reducing DNN Labelling Cost using Surprise Adequacy: An Industrial Case Study for Autonomous Driving. In *Proceedings of ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE Industry Track) (ESEC/FSE 2020)*. 1466–1476.

Jin K. Kim, Michael Chua, Mandy Rickard, and Armando Lorenzo. 2023. ChatGPT and Large Language Model (LLM) chatbots: The current state of acceptability and a proposal for guidelines on utilization in academic medicine. *Journal of Pediatric Urology* 19, 5 (2023), 598–604.

Seah Kim and Shin Yoo. 2021. Multimodal Surprise Adequacy Analysis of Inputs for Natural Language Processing DNN Models. In *Proceedings of the 2nd ACM/IEEE International Conference on Automated Software Testing (AST 2021)*.

Jannik Kossen, Jiatong Han, Muhammed Razzak, Lisa Schut, Shreshth Malik, et al. 2024. Semantic entropy probes: Robust and cheap hallucination detection in llms. *arXiv preprint arXiv:2406.15927* (2024).

LangChain. 2024. LangChain Prompt Template. [https://python.langchain.com/docs/concepts/prompt\\_templates/](https://python.langchain.com/docs/concepts/prompt_templates/).

LangChain. 2024. LangSmith. <https://docs.smith.langchain.com/evaluation/concepts>.

Hugo Laurençon, Lucile Saulnier, Thomas Wang, Christopher Akiki, Albert Villanova del Moral, et al. 2023. The BigScience ROOTS Corpus: A 1.6TB Composite Multilingual Dataset. arXiv:cs.CL/2303.03915 <https://arxiv.org/abs/2303.03915>

Hokyung Lee, Sumanyu Sharma, and Bing Hu. 2024. Bug in the code stack: Can llms find bugs in large python code stacks. *arXiv preprint arXiv:2406.15325* (2024).

Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. 2018. A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks. In *Advances in Neural Information Processing Systems (NeurIPS 2018)*. <https://papers.neurips.cc/paper/7947-a-simple-unified-framework-for-detecting-out-of-distribution-samples-and-adversarial-attacks.pdf> Also available as arXiv:1807.03888.

Lei Ma, Felix Juefei-Xu, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, et al. 2018. DeepGauge: Multi-granularity Testing Criteria for Deep Learning Systems. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering (ASE 2018)*. ACM, New York, NY, USA, 120–131.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, Nov (2008), 2579–2605.

Potsawee Manakul, Adian Liusie, and Mark JF Gales. 2023. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. *arXiv preprint arXiv:2303.08896* (2023).

Roger Mitton et al. 1980. Birkbeck spelling error corpus. *Oxford Text Archive Legacy Collection* (1980).

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, et al. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. 2017. DeepXplore: Automated Whitebox Testing of Deep Learning Systems. In *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP 2017)*. 1–18. <https://doi.org/10.1145/3132747.3132785>

PromptFoo. 2024. PromptFoo. <https://github.com/promptfoo/promptfoo>.

Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, et al. 2021. A survey of deep active learning. *ACM computing surveys (CSUR)* 54, 9 (2021), 1–40.

Pat Rondon, Renyao Wei, José Cambrónero, Jürgen Cito, Aaron Sun, et al. 2025. Evaluating Agent-Based Program Repair at Google. In *2025 IEEE/ACM 47th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. 365–376. <https://doi.org/10.1109/ICSE-SEIP66354.2025.00038>

Tobias Schnabel and Jennifer Neville. 2024. Symbolic prompt program search: A structure-aware approach to efficient compile-time prompt optimization. *arXiv preprint arXiv:2404.02319* (2024).

Reshabh K Sharma, Jonathan De Halleux, Shraddha Barke, and Benjamin Zorn. 2025. PromptPex: Automatic Test Generation for Language Model Prompts. *arXiv preprint arXiv:2503.05070* (2025).

Natalia Silveira, Timothy Dozat, Marie-Catherine de Marneffe, Samuel Bowman, Miriam Connor, et al. 2014. A Gold Standard Dependency Corpus for English. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*.

Ben Snyder, Marius Moisescu, and Muhammad Bilal Zafar. 2024. On Early Detection of Hallucinations in Factual Question Answering. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2024, Barcelona, Spain, August 25-29, 2024*, Ricardo Baeza-Yates and Francesco Bonchi (Eds.). ACM, 2721–2732. <https://doi.org/10.1145/3637528.3671796>

Weihang Su, Changyue Wang, Qingyao Ai, Yiran Hu, Zhijing Wu, et al. 2024. Unsupervised Real-Time Hallucination Detection based on the Internal States of Large Language Models. In *Findings of the Association for Computational Linguistics: ACL 2024*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, Bangkok, Thailand, 14379–14391. <https://doi.org/10.18653/v1/2024.findings-acl.854>

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, et al. 2024. Gemma 2: Improving Open Language Models at a Practical Size. *arXiv:cs.CL/2408.00118* <https://arxiv.org/abs/2408.00118>

Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. 2018. DeepTest: Automated Testing of Deep-neural-network-driven Autonomous Cars. In *Proceedings of the 40th International Conference on Software Engineering (ICSE '18)*. ACM, New York, NY, USA, 303–314.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, et al. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, et al. 2023. Self-Consistency Improves Chain of Thought Reasoning in Language Models. *CoRR* abs/2203.11171 (2023).

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, et al. 2024. SWE-agent: Agent-Computer Interfaces Enable Automated Software Engineering.

Juyeon Yoon, Robert Feldt, and Shin Yoo. 2024. Intent-Driven Mobile GUI Testing with Autonomous Large Language Model Agents. In *Proceedings of the 16th IEEE International Conference on Software Testing, Verification and Validation (ICST 2024)*. 129–139.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of the 29th International Conference on Neural Information Processing Systems - Volume 1 (NIPS'15)*. MIT Press, Cambridge, MA, USA, 649–657.

Yuntong Zhang, Haifeng Ruan, Zhiyu Fan, and Abhik Roychoudhury. 2024. AutoCodeRover: Autonomous Program Improvement. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2024)*. Association for Computing Machinery, New York, NY, USA, 1592–1604. <https://doi.org/10.1145/3650212.3680384>

Shide Zhou, Tianlin Li, Kailong Wang, Yihao Huang, Ling Shi, et al. 2025. Understanding the Effectiveness of Coverage Criteria for Large Language Models: A Special Angle from Jailbreak Attacks . In *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)*. IEEE Computer Society, Los Alamitos, CA, USA, 730–742. <https://doi.org/10.1109/ICSE55347.2025.00209>