Understanding or Imitation? Auditing Conceptual Understanding and Reasoning in Large Language Models

Muhammad Saram Hassan
Lahore University of Management Sciences
Lahore, Pakistan
26100197@lums.edu.pk

Essa Jan
Brown University
Providence, USA
essa_jan@brown.edu

Ramneet Kaur
Computer Science Lab, SRI
Menlo Park, USA
ramneet.kaur@sri.com

Eric Yeh
Artificial Intelligence Center, SRI
Menlo Park, USA
eric.yeh@sri.com

Fareed Zaffar
Lahore University of Management Sciences
Lahore, Pakistan
fareed.zaffar@lums.edu.pk

Ashish Gehani
Computer Science Lab, SRI
Menlo Park, USA
ashish.gehani@sri.com

Abstract

Do large language models genuinely understand concepts, or do they merely reproduce patterns that resemble understanding? As benchmarks increasingly serve as proxies for conceptual competence, it is critical to examine whether high performance reflects coherent reasoning or superficial imitation. We audit benchmark-based claims about conceptual understanding in LLMs through a large-scale reproducibility and extension effort conducted in the context of "Potemkin Understanding in Large Language Models" study. We replicate evaluation pipelines that test whether models can define concepts and apply them consistently across tasks. While we confirm that many standard non-reasoning models exhibit gaps between definition accuracy and downstream application, we find that these effects are substantially less stable and more methodologically fragile than reported.

The Potemkin rate is defined as the frequency with which a model correctly defines a concept but fails to apply it correctly in downstream tasks. Across five identical runs per model, the reported potemkin scores vary by as much as 31%, large enough to reverse qualitative conclusions and revealing significant stochastic instability driven by undocumented randomness and pipeline assumptions. We identify an implicit capability threshold in automated procedures: smaller models frequently fail to generate coherent sub-questions, causing silent breakdowns or misleading scores. Conditioning evaluation on correct definitions risks conflating memorized recall with genuine conceptual understanding. In addition, the testing pipeline is prompt-dependent and relies on the same model self-judging itself, which can mask consistent but incorrect interpretations and inflate apparent coherence. We further show that domain aggregation masks large reliability differences across auto-gradable and human-annotated domains. Most strikingly, reasoning-focused models reduce incoherence by an order of magnitude, suggesting that apparent conceptual failures depend strongly on inference-time regime rather than model class. Our findings indicate that claims about widespread conceptual incoherence in LLMs are directionally supported but empirically fragile. We provide a reproducibility map and concrete recommendations for building more stable and interpretable evaluations of model understanding. The link to our code repository can be found at https://github.com/SRI-CSL/PotemkinBenchmarkReproducibility.

CCS Concepts

• **Computing methodologies** → **Artificial intelligence**; **Natural language processing**; **Natural language generation**;

Keywords

Reproducibility, Benchmark Evaluation, Conceptual Understanding, LLM Evaluation, Replicability, Potemkin

ACM Reference Format:
Muhammad Saram Hassan, Essa Jan, Ramneet Kaur, Eric Yeh, Fareed Zaffar, and Ashish Gehani. 2026. Understanding or Imitation? Auditing Conceptual Understanding and Reasoning in Large Language Models. In *ACM Conference on Reproducibility and Replicability (ACM REP '26), July 20–22, 2026, Delft, Netherlands.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3820002.3828594

Overview of Reproducibility Efforts and Contributions

This paper is written and structured to be an experience paper.
We document what is reproducible, what is fragile, what surprises us, and what the community should do differently. The anchor paper (Manoridis et al. 2025) provides code and data for some components; others required reconstruction or extension to the original code repository. We contribute to building a clean, complete reproducibility map, extended experiments to newer models, and noting down lessons that generalise beyond this specific paper.

ACM REP '26, Delft, Netherlands

ACM ISBN 979-8-4007-2778-8/2026/07
https://doi.org/10.1145/3820002.3828594

1 INTRODUCTION

1.1 The Stakes of LLM Evaluation

Large Language Models (LLMs) are increasingly integrated into high-stakes decision-making pipelines across domains such as medicine, law, education and software engineering [21, 26, 27]. Within these contexts, there operates an implicit assumption: when a model generates a superficially valid response, it does so based on a robust conceptual understanding rather than merely exploiting statistical artifacts or surface-level patterns prevalent in its training data [16]. However, this assumption remains critically underexplored.

Factual fabrications, or hallucinations, can often be identified through external knowledge retrieval [9]. However, conceptual hallucinations, or instances where a model demonstrates linguistic fluency with a term while applying the underlying concept incoherently, are significantly harder to detect and pose a far more insidious risk in safety-critical applications.

This trust is largely mediated through benchmarks. Performance on datasets like MMLU [6], BIG-Bench [23] and HumanEval [3] has become the de facto language for claims about an LLM's capability and reliability. A model that scores 90%, for example, on a professional medical exam is implicitly credited with something like "medical understanding". But this inferential leap, from benchmark score to genuine conceptual grasp, rests entirely on an assumption that is almost never stated explicitly, let alone verified.

This tension is formalised by Mancoridis et al. [16] (hereafter referred to as: *the Potemkin paper*), which introduces the concept of a keystone set and a measurable "Potemkin Rate". Keystone here refers to the minimum number of questions needed to identify an individual's correct understanding of a concept completely. A model is known to exhibit potemkin behavior if it passes definition-level tests on a concept but systematically fails at the downstream concept application tasks, exhibiting the appearance of understanding without substance. The paper reports that potemkin behaviour is ubiquitous across all LLMs.

These are strong claims with significant implications for the entire field of LLM evaluation, and if true, invalidate a substantial fraction of existing benchmark results. If overstated or methodologically fragile, however, they risk producing an incorrect picture of model capabilities. The reproducibility of the specific empirical findings is one question. The robustness of the underlying methodology is another. The generalizability of the results to models released after the study being the third.

This paper is a large-scale reproduction and extension effort that addresses these three dimensions by targeting the Potemkin paper's two primary experimental procedures. **Procedure 1** constructs a curated benchmark that spans 32 concepts in three domains (literary techniques, game theory, and psychological biases) and measures Potemkin rates through a combination of human expert annotation and automated grading. **Procedure 2** is fully automated as it samples questions from the standard NLP benchmarks (MMLU [6] and BIG-Bench Hard [11], uses the LLM to generate and grade concept-application sub-questions, and reports a conservative lower bound on the Potemkin behaviour without requiring manual annotation. Table 1 summarizes the key reproducibility characteristics of both the procedures, which directly shape our methodology (Section 4).

Property	Procedure 1 (Benchmark)	Procedure 2 (Automated)
Scope	3 domains, 32 concepts	Any MMLU / BBH concept
Human annotation required?	Yes (significant expert effort)	No
Bound type	Direct measurement	Lower bound
Code available?	Partial	Full
Scalable to new models?	No, expert-intensive	Yes
Judge bias risk?	Author/expert labels	Same-model self-judging

Table 1: Comparison of reproducibility characteristics across the two Potemkin procedures.

1.2 Contributions

This paper operationalizes our investigation of the Potemkin paper into four specific research questions:

- **RQ1:** Can the original empirical results be reproduced using the provided code and data?
- **RQ2:** Which components required reconstruction, and what did that process reveal about the original methodology?
- **RQ3:** Do potemkin rates hold, increase, or decrease on models untested or released after the original study?
- **RQ4:** What are the conceptual and methodological limitations of the potemkin framework as a diagnostic tool?

Our contributions are threefold:

- **Reproduction:** We attempt exact reproduction of the automated lower-bound procedure (Procedure 2) and partial reproduction of the benchmark procedure (Procedure 1). Full independent replication of Procedure 1 is not feasible: no executable inference code was released, the annotation pipeline depends on expert annotators recruited through Upwork, and author-in-the-loop adjudication of borderline cases cannot be reconstructed from published artifacts alone. We, therefore, treat the released labels as ground truth, verify their arithmetic consistency against the paper's reported values, and independently reconstruct the inference pipeline from the paper's description to approximate the original experimental conditions.
- **Extension:** We extend both procedures to models not included in the original study, including smaller open-source models, newer reasoning models, and quantized variants.
- **Critique:** We identify and systematically test four sources of methodological fragility: stochastic instability across runs, capability-threshold failures in smaller models, same model self-judging circularity, and the keystone validity.

2 BACKGROUND

The Potemkin paper [16] addresses a foundational question in LLM evaluation: when a model answers benchmark questions correctly, does this reflect genuine conceptual understanding, or superficial pattern matching? The paper introduces a formal framework

demonstrating that standard benchmarks (designed to test human understanding) are valid tests for LLMs only if LLMs misunderstand concepts in the same structured ways that humans do.

Theoretical Framework. Conceptual understanding is formalized through the notion of a *keystone set* based on the following components:

- **The Concept Space ($\mathcal{X}$):** Let $\mathcal{X}$ represent the universe of all possible strings or information related to a specific concept, including all potential definitions, examples, and edge cases.
- **Interpretations (f):** An individual's *interpretation* of a concept is defined as a function $f : \mathcal{X} \to \{0, 1\}$. For any given example, this function outputs 1 if the individual believes it is a valid use of the concept, and 0 if they do not. The objectively correct interpretation is denoted as f^*.
- **Boundaries of Misunderstanding ($\mathcal{F}_h$ and $\mathcal{F}_l$):** Errors in reasoning are rarely entirely random. The authors define $\mathcal{F}_h$ as the set of possible human interpretations (predictable ways humans might interpret or misinterpret a concept). Similarly, $\mathcal{F}_l$ represents the set of possible interpretations made by Large Language Models (LLMs).

Definition 2.1 (Keystone Set ($\mathcal{S}$)). A **keystone set**, denoted as $\mathcal{S} \subseteq \mathcal{X}$, is a minimal, highly targeted subset of examples drawn from the concept space. It functions as a diagnostic test with a specific guarantee: if a human correctly categorizes every example within this subset, they must possess the correct overall understanding of the concept.

Intuitively, if a human's interpretation f belongs to the set of possible human interpretations $\mathcal{F}_h$, and their answers match the correct answers for the keystone set ($f(x) = f^*(x)$ for all $x \in \mathcal{S}$), then their overall understanding is perfectly aligned with the truth ($f = f^*$).

Mechanism of Action. This approach is effective because the space of human misconceptions $\mathcal{F}_h$ is relatively small and highly structured. Because people tend to fall into predictable cognitive traps, it is not necessary to test someone on every possible scenario. A carefully curated "keystone" set of questions is sufficient to rule out common errors and definitively certify correct understanding in individuals.

Definition 2.2 (Potemkin Understanding). An LLM has *potemkin understanding* of a concept if $f(x) = f^*(x)$ for all $x \in \mathcal{S}$, i.e., it passes the keystone set defined on the concept. But the LLM does not have the correct interpretation of the concept, .i.e, $f \neq f^*$. Any instance y where $f(y) \neq f^*(y)$ is called a **potemkin**.

In other words, a model exhibits potemkin understanding when it correctly answers keystone questions such as providing an accurate definition but then fails to correctly *apply* the same concept in downstream tasks. This failure mode is possible for LLMs precisely because $\mathcal{F}_l \neq \mathcal{F}_h$: LLMs can misunderstand concepts in non-human ways that happen to produce correct answers on keystone questions through memorisation or pattern matching, while failing on structurally related use tasks such as classification or editing ones. Benchmarks designed for humans are valid tests for LLMs only under the assumption that this gap does not exist. There are 2 main procedures outlined in the paper that measure this Potemkin rate:

Procedure 1: The Benchmark (Direct Measurement). To empirically measure the prevalence of potemkin behaviour, the paper constructs a custom benchmark spanning **32 concepts** across three domains:

- **Literary techniques** (e.g., *analogy*, *ABAB rhyme scheme*, *slant rhyme*): a formal, rule-governed domain. Annotations were produced by the paper's own authors with evaluation on the generation tasks as partially auto-gradable.
- **Game theory** (e.g., *Nash equilibrium*, *strict dominance*, *Pareto optimality*): a formally specified domain with unambiguous correct answers. Instances were produced by Economics PhD students and graded automatically.
- **Psychological biases** (e.g., *sunk cost fallacy*, *anchoring bias*): a subjective domain requiring expert judgment. 40 Reddit text responses per concept were gathered from `r/AmIOverreacting` and annotated by behavioural scientists recruited via Upwork; these annotations are not independently replicable.

For each of the 32 concepts, seven models were evaluated on four task types: one *Keystone* and three *Use* tasks.

KEYSTONE TASK:

(1) **Definition** The model is prompted to define the concept in the selected domain. A correct definition constitutes passing the keystone and is the conditioning variable for all the downstream evaluation: only model-concept pairs where the model provides a correct definition are scored on the 3 use tasks mentioned below. Across all models and concepts, 94.2% of definitions were judged correct which is a suspiciously high rate, and one that conditions the entire downstream analysis on an easily-faked signal.

USE TASKS:

(2) **Classification:** Given an example, the model must determine whether it is a valid instance of the concept (Yes/No). Labels are compared against gold annotations; game theory labels are auto-graded, while literary and psychological bias labels are human-annotated.
(3) **Generation (Constrained):** The model must produce an example of the concept that adheres to domain-specific constraints. For instance, for *strict dominance* in game theory, the model must construct a payoff matrix satisfying the formal definition; for *ABAB rhyme scheme*, it must produce a poem in which first and third lines rhyme, and second and forth lines rhyme.
(4) **Editing:** The model is presented with a partially correct instance and must modify it to either satisfy or violate the concept. For instance, a partially completed haiku may be presented with a missing word, and the model must fill in a word that preserves the syllable structure.

Definition 2.3. The **Potemkin rate** (PR) for a task is defined as the proportion of use task questions answered *incorrectly*, conditional on a correct concept definition by the model. Since the chance accuracy on classification tasks is 0.5, the PR is scaled so that the chance-level performance maps to PR = 1 and perfect performance maps to PR = 0 resulting in the formula:

$$\mathrm{PR} = 2(1 - \bar{A}), \tag{1}$$

where $\bar{A}$ is mean accuracy on use tasks, conditioned on a correct definition.

Across 7 models and 32 concepts, the paper reports average potemkin rates of **0.55** (classify), **0.40** (generate), and **0.40** (edit), averaged across all models confirming the widespread potemkin behaviour even among frontier models.

Procedure 2: The Automated Lower Bound. The second procedure is fully automated and generalises beyond the custom benchmark by using questions from two standard NLP benchmarks: **Massive Multitask Language Understanding** (MMLU) [6], a 57-subject multiple-choice dataset spanning STEM, humanities, and social sciences, and **BIG-Bench Hard** (BBH) [23], a suite of 23 tasks selected to be challenging for LLMs. For each trial, the pipeline operates as follows:

(1) A benchmark question is sampled uniformly at random from MMLU or BBH.
(2) The *responder* model (the LLM under test) determines whether the question tests a specific concept via a concept-detection prompt. If no concept is detected, the question is discarded and a new one is sampled.
(3) The responder answers the benchmark question. If the answer is incorrect against the gold label, the trial is discarded. Hence, only correctly-answered questions proceed, ensuring the keystone condition is met.
(4) The responder generates $m = 5$ subquestions that probe the same underlying concept from different angles (i.e., use tasks) randomly. If fewer than 5 subquestions are produced, the generation is retried up to 10 times.
(5) For each subquestion, the responder produces a correct answer and an *error-injected* variant (a subtly incorrect answer that a knowledgeable person would recognise as wrong). The *same model* then acts as judge, grading both versions: it should mark the original as correct and the error-injected version as incorrect. Disagreement indicates incoherence.

Coherence $\bar{C}$ is the fraction of trials in which the judge grades both versions correctly. The **potemkin rate** for Procedure 2 is:

$$\mathrm{PR} = 2(1 - \bar{C}) \tag{2}$$

This formula uses the same scaling as Equation 1: PR = 0 indicates perfect coherence and PR = 1 indicates random chance. Critically, the formula has no enforced upper bound. If $\bar{C} < 0.5$, it means the model is *anti-coherent*, grading correct answers as incorrect more often than chance makes the potemkin rate exceeds 1.0. This property is not explicitly acknowledged in the original paper and becomes consequential in our reproduction.

Separately, the paper also measures **incoherence** as a diagnostic: the model is prompted to *generate* an instance of a concept, then in a separate query asked to *classify* whether that same output is a valid instance. The fraction of cases where the model disagrees with its own generation is the incoherence score. This captures a different failure mode - internal representational inconsistency - and is reported alongside the PR lower bound.

The overall reported automated PR and incoherence are **0.62** and **0.22** respectively, across 8 models. A notable outlier is *o3-mini*, which achieves an incoherence of just 0.03 versus GPT-4o's 0.64, suggesting that reasoning-focused inference regimes substantially reduce potemkin behaviour.

Table 1 summarises the key reproducibility characteristics of both procedures, which directly shape our approach in Section 4.

3 RELATED WORK

The call to closely analyze the ability of machine-learning trained systems to reason correctly has been a concern for years [4]. With the rise of highly performant LLMs displaying human-like abilities to reason across a broad range of tasks and environments, there has been growth in works scrutinizing and questioning the ability of current generation LLMs to actually reason [2]. This is of particular concern in the area of AI alignment, where systems may need to perform complex inferences to determine if their behavior in what amounts to open-world domains meets expectations and stays within safety constraints [1].

Chief amongst many concerns is the tendency of machine learning systems to learn *shortcuts*, surface feature cues in the input that directly correlate with the answer. Shortcuts give the impression that the system is capable of complex conceptual understanding, but is limited in practice by the domain of the shortcut and thus does not generalize. It has been a long-running concern for machine learning systems [4], especially for systems performing natural language understanding (NLU) tasks [5]. NLU systems are expected to perform complex reasoning over inputs and generate answers in natural language, such as answering natural language questions. The area has driven the requirement of benchmarks for assessing LLMs [7, 19, 25]. In particular, the consistency and coherence of question-answering systems have been examined under conceptual variance and logical consistency [20].

Prior work has measured the ability of LLMs to perform reasoning against hierarchical models of task complexity, most notably Bloom's Taxonomy [12] – a framework originally developed for pedagogy where problems are mapped into a hierarchy of increasingly difficult problem types organized around the type of cognitive ability required. Bloom-based taxonomies can analyse an LLM's performance on question-answer benchmarks [10] and medical examination [8]; showcasing that these models tend to perform poorly on higher-order reasoning tasks.

Concept inventories are the sets of questions that measure the level of concept understanding by students. These inventories are widely used in educational research for assessing students' understanding across multiple domains [22]. They are also equivalent to the keystone set described in [16]. However, developing these inventories of questions is domain-specific, and there is significant variance in the development, validation, and reliability of such inventories [17]. Indeed, establishing psychometric validity and reliability (measurement of mental processes) is critical for education and is central to our recommendations for assessments of LLMs and other automated systems: If an instrument fails to generate reliable measures of a phenomenon in question, it compromises the ability to make concrete, trustworthy statements about that phenomenon.

Using the same LLM to generate and evaluate hypotheses is a popular and widespread method for automatically scoring complex hypotheses produced by LLMs [28]. However, using the same LLM to both generate and evaluate can be problematic, as LLMs have

been shown to display self-preference bias and score their own outputs higher than those from other models [15, 18]. Preference bias has also been demonstrated for models within the same family and between a source and student [14], and AI models have even been found to favor AI-created outputs over human outputs [13].

4 REPRODUCIBILITY ANALYSIS: SETUP & METHODOLOGY

This section provides the complete specification of our reproduction and extension efforts, detailed to the level required for independent replication. To our knowledge, no published reproducibility study has targeted the Potemkin paper [16] prior to this work. The paper was accepted at ICML 2025; code was made available via GitHub of the `PotemkinBenchmark` repository.[1]

As described in Section 2, Procedure 1's annotation pipeline relies on expert annotators recruited through Upwork and is not independently replicable without significant cost; no executable code is provided to reconstruct its results. Procedure 2's code is available and fully automated, making it the primary target for our reproduction effort (refer to Table 1 for a summary of differences).

4.1 Experimental Setup

The experiments in the Potemkin paper primarily relied on API-accessed models and, therefore, did not require local GPU inference. We conducted all assessments on GPUs, both for inferences on the larger open-source models `Llama-3.3-70B` and `Qwen2-VL-72B` from the Potemkin paper as well as the additional experiments introduced in this paper with smaller open-weight models – `Llama-3.1-8B`, `Mistral-7B-v0.3`, and `Qwen-2.5` variants. All reproduction and additional experiments were performed on a Linux workstation equipped with RTX 6000 GPUs. The software environment used PyTorch together with the Hugging Face `transformers` library for local inferences. Details on the experimental setup are listed in Table 11 (in Appendix).

Additionally, to evaluate model outputs, we employed `gpt-5.4-mini` as an automated LLM judge. The judge was accessed via the OpenAI API and prompted to return a strictly binary verdict (1 or -1) to determine task correctness. To ensure deterministic and reproducible grading, the sampling temperature was set to 0, and the maximum retry limit for API calls was set to 5.

4.2 Artifact Availability and Assessment

We cloned the `PotemkinBenchmark` repository at commit 39, which is the latest commit at the time of the original paper's submission. The two procedures differ substantially in terms of artifact completeness, necessitating separate treatment, as described below.

4.2.1 Procedure 1: The Benchmark Framework.

Available Artifacts. The repository provides: (a) a raw question-only dataset (`BenchmarkDataset`) containing all classification, generation, and editing instances across the three domains; (b) a pre-computed `labeled_instances.csv` file containing model responses and human-produced annotations for all 3,159 instances; (c) the `potemkin_rates.py` scoring script; and (d) prompt templates for definition elicitation and task evaluation. No executable code for model inference, response collection, or annotation pipeline orchestration is provided.

Label Verification versus Experimental Reproduction. Relying solely on the repository restricts us to running `potemkin_rates.py` against the published `labeled_instances.csv`. While this confirms arithmetic consistency with Table 1 of the original paper (see our Table 3), it constitutes *computational label verification* rather than experimental reproduction. We cannot verify whether querying the same models today would yield comparable responses, nor can we assess the stability of the annotation decisions embedded in `labeled_instances.csv`.

To bridge this gap, we independently reconstructed the inference and evaluation pipeline from the paper's methodological descriptions. Although this falls short of true experimental replication, it establishes the most defensible baseline available: we confirm arithmetic correctness, document precisely what was and was not released, and test current models under the same prompting methodology. Any deviation from the original results constitutes a reproducibility finding in its own right.

Annotation Dependencies and Replicability Boundaries. The three domains differ substantially in their annotation methods, creating asymmetric replicability profiles summarised in Table 2. We formally define a *high* replicability boundary as fully automated and deterministic, *medium* as dependent on authors' internal validation, and *hard* as reliant on anonymous third-party annotators without published individual judgments. Game theory uses deterministic evaluation functions for generation and editing tasks, making it the most reproducible domain. Literary techniques labels were produced entirely by the paper's own authors — a softer boundary: the annotations are plausible but lack independent validation. Psychological biases labels were produced by behavioural scientists recruited via Upwork; their individual judgments are not published, no per-annotator reliability statistics are provided, and the annotator pool is not reconstructible. We treat this as a *hard replicability boundary* for that domain.

We designed an LLM-as-judge pipeline to partially overcome the human annotation dependency (Section 5.2), substituting automated evaluation for domain experts.

Computational Verification. Procedure 1 results are computationally verifiable. We executed `potemkin_rates.py` against the published `labeled_instances.csv` without modification. Every one of the 24 per-model-per-task (derived from the specific models and tasks evaluated in the original study's pre-computed dataset) Potemkin rate values matches the paper's Table 1 within 2× the reported standard error, with a maximum absolute deviation of 0.006. This confirms that the code and data together reproduce the paper's numbers exactly.

Non-replicability. However, Procedure 1 is **not independently replicable**. As pointed out previously, the annotation pipeline depends on resources unavailable to independent researchers for annotations. We treat this as an inherent limitation of human-annotated benchmarks rather than a deficiency of this specific paper, but note

[1] https://github.com/MarinaMancoridis/PotemkinBenchmark

Domain	Annotation Method	Auto-Gradable	Replicability
Game Theory	Economics PhD students (classification); deterministic functions (generation, editing)	Mostly	High
Literary Techniques	Paper authors (all tasks)	Partial	Medium
Psychological Biases	Behavioural scientists via Upwork (all tasks)	No	Hard boundary

Table 2: Annotation methods and replicability status across Procedure 1 domains. A *hard boundary* indicates that the annotator pool and individual label decisions are not recoverable from public artifacts.

that the original work does not explicitly acknowledge this boundary. We therefore accept the released labels as ground truth and focus our independent replication effort on Procedure 2.

4.2.2 *Procedure 2: Automated Framework.*

Available Artifacts. The repository provides a complete, executable automated pipeline: all Python source files (`main.py`, `utils.py`, `run_experiments.py`), prompt templates, and grading logic. The MMLU and BBH benchmark datasets are accessed via the Hugging Face `datasets` library and require no additional data collection. No human annotation is required at any stage.

Assessment. The pipeline is independently runnable with minimal setup — install dependencies from `requirements.txt`, supply an API key or a local model path, and execute `main.py`. We successfully ran it without reconstructing any core components. However, several undocumented behaviours caused unstable results for models outside the original study's scope. These required targeted modifications before reliable reproduction was achievable are described in the following Section 4.3.

4.3 Pipeline Modifications and Extensions

This section documents all changes made to the original codebase for each procedure, along with a rationale for each change. After applying these modifications, both procedures are end-to-end runnable from the published question sets. The steps required to reproduce our results are summarised in the README file of our repository.

4.3.1 *Procedure 1:*

Inference Pipeline Reconstruction. No executable code was provided for Procedure 1. We reconstructed the evaluation pipeline from scratch following the paper's methodology and applied it to the published `BenchmarkDataset`. The reconstructed pipeline uses the same prompt templates and task definitions described in the paper. For tasks that are not auto-gradable (literary techniques and psychological biases), model responses are evaluated by `gpt-4o-mini` acting as judge, prompted with task-specific rubrics mirroring the original annotation criteria. All specific prompt templates have been preserved and are available in Appendix B.1 as well as the accompanying code repository.

4.3.2 *Procedure 2:*

a) Extension to open-source and reasoning models. To broaden the scope of our evaluation beyond proprietary APIs, we extended our analysis to encompass a variety of smaller, open-source architectures as well as specialized reasoning models. These models were deployed and executed locally on dedicated GPUs. This localized inference setup not only ensured full control and reproducibility of our experimental conditions, but it also allowed us to thoroughly evaluate the performance of these architectures without the constraints of API rate limits or network latency.

b) Smaller model instruction following. Smaller models frequently produced fewer than five subquestions on the first attempt. We introduced a retry loop that re-prompts up to three times before falling back to all available output, with explicit logging of any shortfall. The `relies_on_concept` filter also failed silently when smaller models included the token `YES` inside otherwise incoherent responses; we added explicit parsing rules to record such cases as evaluation failures rather than valid responses.

c) Subquestion generation cap & Output persistence. The original pipeline caps subquestion generation at exactly five: if a model produces more, the entire generation call is restarted, introducing additional non-determinism and unnecessary API cost. We modified this by randomly sampling five questions whenever more than five are produced, eliminating the restart loop. By taking a uniform random sample of 5 questions, we avoid divergence from the intended distribution while eliminating the costly restart loop. We also implemented automated saving of per-run Potemkin rate scores to structured CSV output, absent from the original release.

5 REPRODUCIBILITY RESULTS

5.1 Procedure 2: Automated Evaluation

This section presents results from Procedure 2, the fully automated coherence pipeline. Experiments are organised into four components: reproduction on the original paper's frontier models to establish a baseline; extension to smaller models to probe capability thresholds; an analysis of reasoning models testing our core hypothesis that chain-of-thought inference reduces Potemkin behaviour; and a paired self-judging experiment examining how model scale mediates the circularity problem.

5.1.1 *Reproducibility on Originally Reported Models.*

We begin by reporting our reproduction results for Procedure 2 on two models from the original study: `Llama-3.3-70B-Instruct` and `Qwen2-VL-72B-Instruct`. We ran five independent iterations on both MMLU and BBH for each model, using the modified pipeline described in Section 4.3. Each iteration samples a fresh random subset of

Task Type	Original ($\mu \pm$ SE)	Our Verification	Max Deviation	Status
Classify	0.55 ± 0.02	0.545 ± 0.021	0.006	Verified
Generate	0.40 ± 0.03	0.401 ± 0.033	0.005	Verified
Edit	0.40 ± 0.02	0.401 ± 0.018	0.004	Verified

Table 3: Procedure 1's computational verification. All 24 per-model-per-task values fall within 2× standard error of the published values. "Verified" denotes arithmetic consistency with `labeled_instances.csv`, *not* independent replication of model outputs or human annotations.

questions, which in turn triggers fresh concept detection, subquestion generation, and coherence grading compounding stochasticity across multiple pipeline stages. A critical ambiguity is that the Potemkin paper does not specify which benchmark dataset (MMLU or BBH) underlies the Potemkin rates reported in its Table 2; we therefore run and report results on both datasets in our paper (see Table 4).

Table 4 reveals several findings. First, the run-to-run variance is substantial even for large frontier models: Llama-3.3-70B on MMLU ranges from 0.59 to 0.83 (spread of 0.24), and Qwen2-VL-72B on MMLU ranges from 0.28 to 0.79 (spread of 0.51). These are not minor fluctuations since a single lucky run could place Qwen2-VL-72B at 0.28, which would suggest near-perfect coherence, while another places it at 0.79, suggesting a severe potemkin behaviour.

Second, our mean values diverge from the original paper's single reported number on Potemkin Rate (PR). The paper reports a Potemkin rate of 0.82 for both Llama-3.3-70B and Qwen2-VL-72B. Our Llama-3.3-70B averages 0.73 on MMLU and 0.69 on BBH; our Qwen2-VL-72B averages 0.57 on MMLU and 0.74 on BBH. Multiple factors contribute to this gap:

(1) **Dataset ambiguity.** The original paper does not specify whether the reported numbers come from MMLU, BBH, or a combination. Our results show that the choice of benchmark matters: Qwen2-VL-72B differs by 0.17 between MMLU and BBH means.
(2) **Question sampling.** Each run samples different questions from the benchmark, and each question tests a different concept. Since the pipeline then generates *new* subquestions for each sampled concept, the randomness compounds: different questions → different concepts → different subquestions → different answers → different coherence judgments.
(3) **Stochastic model output.** Even after fixing the questions, the model's response varies across calls due to temperature and sampling. The judge's grading of those responses introduces yet another layer of randomness.

The compounding of these sources of variance means that single-run point estimates, as reported in the original paper, are insufficient to characterise a model's true Potemkin rate. Our five-run analysis suggests that the original paper's numbers fall within the range of plausible outcomes but do not represent stable measurements.

5.1.2 Extension to Smaller Models. We extend Procedure 2 to smaller open-source models to test whether the framework generalises beyond the frontier models examined in the original study. Table 5 presents five-iteration results for models ranging from 0.5B to 8B parameters.

Three findings emerge. First, the run-to-run variance observed in frontier models (Section 5.1.1) is replicated - and in some cases amplified - in smaller models. `Llama-3.1-8B` on MMLU spans 0.80 to 1.11 across runs (spread of 0.31); `Qwen2.5-7B` on MMLU spans 0.74 to 1.05 (spread of 0.31). These differences are large enough to change the qualitative interpretation of whether a model is performing above or below chance. As established in Section 2 (Equation 1), a PR below 1.0 indicates above-chance coherence, a PR of 1.0 indicates chance-level performance, and a PR above 1.0 indicates anti-coherent behaviour - a distinction that matters when individual runs cross this threshold.

Second, several model-benchmark configurations produce mean Potemkin rates exceeding 1.0 (Llama-3.1-8B BBH: 1.02, Llama-3.2-3B MMLU: 1.02, Qwen2.5-0.5B MMLU: 1.12). As discussed in Section 6, this is a property of the formula that the original paper never acknowledges. We hypothesize that these inflated, out-of-bounds rates are driven by three primary compounding failures within the evaluation pipeline: (1) a fundamental misunderstanding of the target concepts by the model, which leads to the generation of poorly formulated or entirely irrelevant subquestions; (2) the subsequent generation of irrelevant, random, or hallucinated answers to those subquestions; and (3) inadequacies in the automated judge, which may produce false positives by erroneously validating flawed reasoning steps, thereby artificially inflating the metric.

Third, a clear capability threshold emerges. Qwen2.5-0.5B failed to generate any valid subquestions in all five BBH runs and produced usable output in only 3 of 5 MMLU runs. Even the completed MMLU runs show extreme variance (1.12 ± 0.23). The pipeline presupposes that the model can reliably generate coherent subquestions from benchmark-style inputs but models below approximately 3B parameters cannot satisfy this precondition. Smaller models also struggle with *instruction following*: they frequently fail to format outputs as required by the pipeline (e.g., generating freeform text instead of structured subquestions), a behavior observed uniformly across the tested benchmarks rather than being tied to specific tasks, triggering parsing failures that manifest as silent data loss rather than explicit errors. The original paper restricts itself to seven large frontier models and never acknowledges this implicit capability assumption.

5.1.3 Extension to Reasoning Models. The original paper includes results on `o3-mini` and `GPT-o1-mini` for Procedure 2, reporting a striking contrast in incoherence between reasoning and non-reasoning models (`o3-mini`: 0.03 versus `GPT-4o`: 0.64), but provides

Model	Data	R1	R2	R3	R4	R5	Mean ± Std
Llama-3.3-70B	MMLU	0.81	0.59	0.83	0.69	0.74	0.73 ± 0.10
Llama-3.3-70B	BBH	0.93	0.59	0.62	0.70	0.60	0.69 ± 0.14
Qwen2-VL-72B	MMLU	0.63	0.79	0.28	0.54	0.60	0.57 ± 0.18
Qwen2-VL-72B	BBH	0.75	0.83	0.65	0.79	0.70	0.74 ± 0.07

Table 4: Procedure 2's reproduction results (Potemkin Rate) for models from the original study. R1–R5 denote five independent runs with different random question samples. Original paper reports: `Llama-3.3-70B` = 0.82 (SE 0.02); `Qwen2-VL-72B` = 0.82 (SE 0.00).

Model	Dataset	R1	R2	R3	R4	R5	Mean ± Std
Llama-3.1-8B	MMLU	1.11 (0.04)	1.05 (0.02)	0.90 (0.04)	0.80 (0.03)	0.90 (0.06)	0.95 ± 0.11
Llama-3.1-8B	BBH	1.09 (0.03)	1.14 (0.04)	0.90 (0.03)	1.09 (0.04)	0.90 (0.03)	1.02 ± 0.10
Llama-3.2-3B	MMLU	1.08 (0.02)	1.04 (0.02)	1.10 (0.04)	0.91 (0.02)	0.96 (0.06)	1.02 ± 0.07
Llama-3.2-3B	BBH	1.06 (0.03)	0.82 (0.03)	1.01 (0.07)	0.87 (0.07)	1.08 (0.03)	0.97 ± 0.10
Qwen2.5-7B	MMLU	0.79 (0.01)	1.05 (0.03)	0.74 (0.02)	0.81 (0.01)	0.92 (0.01)	0.86 ± 0.11
Qwen2.5-7B	BBH	0.72 (0.01)	0.80 (0.02)	0.84 (0.03)	0.73 (0.02)	0.83 (0.03)	0.78 ± 0.05
Qwen2.5-0.5B†	MMLU	0.90 (0.13)	1.44 (0.09)	—	1.03 (0.03)	—	1.12 ± 0.23
Qwen2.5-0.5B†	BBH	FAIL	FAIL	FAIL	FAIL	FAIL	—

Table 5: Potemkin Rate for smaller open-source models across five independent runs (R1–R5). Values shown in brackets for each run represent the standard error within that run. †`Qwen2.5-0.5B`: only 3 of 5 MMLU runs produced valid output (R3 and R5 produced no output); all 5 BBH runs failed with `NoneType` errors due to total subquestion generation failure.

no dedicated analysis of this gap. Our central hypothesis, motivated by this finding, is that chain-of-thought reasoning reduces Potemkin rates by enforcing consistency between a model's stated understanding of a concept and its downstream application. We test this by extending the evaluation to open-source reasoning models.

Firstly, `GPT-OSS 20B` on BBH (mean PR: 0.53 ± 0.21) is starkly lower than the non-reasoning models at comparable scale (`Qwen2.5-7B`: 0.78 ± 0.05 on BBH; `Llama-3.1-8B`: 1.02 ± 0.10). The directional signal is encouraging, though the high run-to-run variance (±0.21, matching the worst frontier models from Section 5.1.1) warrants caution. Table 6 presents the five-iteration stability results for `GPT-OSS 20B` on BBH, matching the format of the smaller-model stability analysis in Section 5.1.2.

Table 7 below reports single-run Potemkin rates for a DeepSeek reasoning model.[2]

Three findings emerge. First, DeepSeek-R1-Distill-14B achieves a Potemkin rate of 0.47 on MMLU - substantially below every non-reasoning model tested at comparable or larger scale. `Llama-3.3-70B` averaged 0.73 on MMLU; `Qwen2-VL-72B` averaged 0.57 (Table 4). A 14B reasoning model outperforming 72B non-reasoning models by this margin despite having a 4-5× parameter disadvantage directly supports the hypothesis that chain-of-thought inference reduces Potemkin behaviour.

[2]Due to the substantial computational resources and time required for inference on this architecture, we were restricted to a single evaluation run. This result is included to provide a directional signal regarding the behavior of reasoning models, rather than a statistically bounded estimate.

Taken together with the original paper's `o3-mini` and `GPT-o1-mini` results, our evidence consistently supports the hypothesis that reasoning reduces Potemkin rates.

5.1.4 External Model as the Judge. The original Procedure 2 uses the same model as both the responder and the judge. This creates a circularity risk: a model that consistently misapplies a concept will also consistently judge its own misapplication as correct, inflating apparent coherence. We designed and executed a cross-model judging experiment to test this hypothesis directly.

Methodology. For a given benchmark question, the *responder* model completes the standard pipeline (concept detection, subquestion generation, answering). Then, instead of the responder grading its own answers, an independent *judge* model evaluates whether each answer is correct or incorrect. This decoupling means the judge may not share the responder's systematic errors, and any consistent misapplication by the responder should appear as incoherence under an external judge.

We tested three configurations on MMLU (10 trials each, seed 42), and report our results in Table 8.

These results reveal a clear pattern. When DeepSeek-R1-Distill-7B judges its own answers, the Potemkin rate is 0.95 (incoherence: 0.455). When the same answers are judged by `Llama-3.1-8B`, the Potemkin rate rises to 1.06 (incoherence: 0.512) - a shift of 0.11 in Potemkin rate. This is consistent with the self-judging circularity hypothesis described in Section 6: the responder's systematic errors are partially masked when it evaluates its own outputs, because both

Model	Dataset	R1	R2	R3	R4	R5	Mean ± Std
GPT-OSS 20B	BBH	0.43 (0.02)	0.51 (0.01)	0.25 (0.03)	0.68 (0.04)	0.77 (0.03)	0.53 ± 0.21
GPT-OSS 20B	MMLU	0.46 (0.02)	0.64 (0.04)	0.57 (0.02)	0.47 (0.04)	0.43 (0.03)	0.51 ± 0.09

Table 6: Potemkin Rate stability results for `GPT-OSS 20B` by Procedure 2 (self-judge).

Model	Dataset	PR (Self-Judge)	*n* runs
DeepSeek-R1-Distill-14B	MMLU	0.47	1

Table 7: Procedure 2 Potemkin rates for DeepSeek reasoning models (self-judge, single run).

the answer and the evaluation share the same misconceptions. An external judge, not sharing those biases, surfaces more incoherence.

Other results can also be seen in the table below. Though acknowledged briefly in the original paper, the practical implication is significant: the self-judging design used in the original paper's Procedure 2 may systematically underestimate or overestimate Potemkin rates as discussed laters. If a model consistently misapplies a concept but does so in a self-consistent way, self-judging registers this as coherent. Cross-model evaluation exposes the misunderstanding. This finding suggests that the original paper's reported Potemkin rates may be *lower bounds on lower bounds* i.e the true extent of potemkin behaviour could be higher than reported in most cases.

Furthermore, `Qwen2.5-7B`'s relatively stable Potemkin rate under cross-judging may be partly explained by its underlying training recipe. Its post-training explicitly targets instruction following, structured outputs, and logical reasoning, which are exactly the intermediate skills required by the Procedure 2 pipeline. This robust instruction-tuning may make `Qwen2.5-7B` less prone to pipeline-induced failures - such as malformed subquestions, poorly structured answers, or incoherent self-evaluation - rather than proving a complete absence of conceptual inconsistency. This interpretation aligns with the model's technical report, where `Qwen2.5-7B-Instruct` outperforms models like `Gemma2-9B` and `Llama-3.1-8B` on most 7B-level instruction-tuned benchmarks, including MMLU-Pro, MATH, and HumanEval [24].

5.2 Procedure 1: Independent Replication

5.2.1 Independent Replication with LLM-as-Judge. Procedure 1 of the original paper uses human expert annotators to label 3,159 concept-application instances. As established in Section 4.2.1, we verified computational reproducibility but cannot independently replicate the human annotation. To bridge this gap, we designed an automated pipeline that replaces human annotators with LLM-as-judge evaluation. We empirically measure this substitution effect in Table 9, showing high alignment with human labels. Furthermore, a random sample of these automated judgments was manually inspected by the authors to confirm alignment with the original dataset.

Methodology. We use the exact questions from the original paper's `BenchmarkDataset` across all three domains (literary techniques, game theory, psychological biases) and all four task types (Define, Classify, Generate, Edit). For each question, a *responder* model produces an answer following the original methodology. A *judge* model then evaluates the answer using task-specific prompts designed to mirror the original annotation rubric:

- **Define**: The judge evaluates whether the model's definition captures the essential properties of the concept.
- **Classify**: Auto-gradeable - we compare the model's Yes/No output directly against the ground-truth label.
- **Generate**: The judge evaluates whether the generated example correctly demonstrates the concept.
- **Edit**: The judge evaluates whether the edited text correctly applies or removes the concept as instructed.

We report results from two judge conditions: a *self-judge* baseline where the responder model evaluates its own outputs, and an *external judge* condition using `gpt-5.4-mini`. We first validate the external judge's reliability against the published human annotations before interpreting the comparative results.

Judge Validation. To assess whether `gpt-5.4-mini` produces reliable labels as a substitute for human annotation, we compared its verdicts against the ground-truth labels in `labeled_instances.csv` on a stratified sample across all four task types (Table 9).

Classify agreement is near-perfect (99.33%) because correctness is determined by binary comparison against ground-truth labels, not LLM judgment. Define agreement (93.18%) is comparable to expected inter-annotator reliability for definitional quality assessment, providing reasonable confidence in the judge's fitness for that task. Generate and Edit show lower but expected agreement (63.98% and 70.81%), consistent with the inherent ambiguity of evaluating open-ended concept application. These rates should be read as a signal about the subjectivity of the tasks, not merely as a limitation of the LLM judge.

Replication Results. Table 10 below presents Potemkin rates for additional four models under both judge conditions, alongside the original paper's human-annotated reference values.

6 DISCUSSION

The Potemkin paper makes a genuine and important contribution to how the field evaluates benchmark validity. The following actionable improvements are offered to make this methodology more durable and replicable.

Document and seed run-to-run variance. Relying on point estimates with standard errors across concepts masks the procedure's stochastic instability. Our five-iteration replication shows that a single run can yield Potemkin rates ranging by 0.20-0.30 on the

Responder	Judge	PR ± SE	Incoherence
DeepSeek-R1-7B	DeepSeek-R1-7B	0.95 ± 0.04	0.455
DeepSeek-R1-7B	Llama-3.1-8B	1.06 ± 0.02	0.512
Qwen2.5-1.5B	Qwen2.5-1.5B	1.301 ± 0.046	0.600
Qwen2.5-1.5B	GPT-5.4-Mini	1.126 ± 0.033	0.582
Qwen2.5-7B	Qwen2.5-7B	0.620 ± 0.025	0.364
Qwen2.5-7B	GPT-5.4-Mini	0.618 ± 0.041	0.337

Table 8: Potemkin Rate and Incoherence results on MMLU (10 trials, seed 42) across different responder and judge configurations.

Task	Agreements	Total	Accuracy
Classify	298	300	99.33%
Define	328	352	93.18%
Edit	114	161	70.81%
Generate	103	161	63.98%
Total	**843**	**974**	**86.55%**

Table 9: `gpt-5.4-mini` judge agreement with human-annotated labels from `labeled_instances.csv`. Classify is auto-graded; near-perfect agreement confirms correct implementation. Lower agreement on Generate (63.98%) and Edit (70.81%) reflects genuine annotation subjectivity in open-ended tasks rather than judge unreliability.

same model and dataset due to variance. Papers should report the standard deviation across N identically seeded runs and explicitly document hidden parameters like the subquestion generation cap ($m = 5$).

Validate the keystone before conditioning on it. Procedure 1 conditions the entire downstream analysis on models providing a correct definition, treating definition accuracy(94.2%) as a certified signal of understanding. If the keystone is trivially easy to pass not through understanding but through memorisation, the framework measures a difficulty asymmetry between definition recall and concept application—not conceptual incoherence. Compounding this, the definition judgments were made by the paper's own authors with no inter-rater reliability reported. Future versions should have a second, independent annotator evaluate a stratified sample and report Cohen's kappa. The conditioning variable for the entire benchmark analysis deserves the same scrutiny as the outcome variables.

Acknowledge the metric's unbounded range. The Potemkin rate formula $2(1 - \bar{C})$ is presented as ranging from 0 (perfect coherence) to 1 (chance), but we observed PR > 1.0 in multiple configurations (Table 5). Values above 1.0 indicate *anti-coherent* behaviour where a judge grades correct answers as incorrect more often than chance. The full metric range should be explicitly defined, and configurations exceeding 1.0 must be contextualized.

Isolate and measure stage-wise pipeline failures. While the primary scope of this reproducibility study was to evaluate the end-to-end Potemkin rate exactly as originally defined, our observations strongly suggest that these inflated, out-of-bounds rates are the result of cascading errors. Because the pipeline consists of discrete, observable steps - concept detection, subquestion generation, and validation - future work should isolate these components and measure the independent error rate at each stage to explicitly verify how these failures compound to inflate the final metric.

Treat self-judging as a baseline, not a design choice. Procedure 2 uses the same model as both responder and judge. A model that consistently misapplies a concept but does so self-consistently will appear coherent under self-judging, because both the answer and the evaluation reflect the same underlying misconception. This conflates two distinct failure modes: *incoherence* (the model contradicts itself) and *consistent misunderstanding* (the model applies a uniformly wrong interpretation). Same-model evaluation should strictly be treated as a preliminary baseline requiring cross-model validation.

Validate LLM-as-Judge capabilities. Although our external judge aligned well with human annotations on objective tasks, its reliance for subjective evaluation introduces unexamined biases. Future work must deeply validate the failure modes of the judge model itself to ensure it does not systematically distort the evaluation.

Report domain-level variance as a primary result, not an appendix table. The three Procedure 1 domains differ substantially in annotation reliability (Table 2): (auto-graded vs. author-annotated vs. external experts). Aggregating them conflates formally specified evidence with subjective interpretation. Domain-stratified Potemkin rates, currently relegated to the appendix, should be the primary reported result.

Surface the reasoning model finding as a central result. Evidence shows reasoning models (`o3-mini`, `DeepSeek-R1-Distill-14B`) drastically reduce incoherence compared to standard models. This critical finding qualifies the claim that Potemkin understanding is ubiquitous across all LLMs, suggesting it may instead be an artifact of standard inference regimes and should be analyzed as such.

For the Broader Research Community. Evaluation methodology papers build the infrastructure for downstream research and must be held to higher reproducibility standards. If baseline methodologies are fragile, every paper that builds upon them inherits that

	Define (Acc.)		Classify (PR)		Generate (PR)		Edit (PR)	
Model	**Self**	**GPT**	**Self**	**GPT**	**Self**	**GPT**	**Self**	**GPT**
Original (human, 7-model avg.)	94.2%		0.55		0.40		0.40	
`Llama-3.1-8B`	66.7%	83.3%	1.105	1.105	0.750	1.607	1.044	1.933
`Mistral-7B-v0.3`	97.6%	83.3%	0.970	0.970	0.026	1.636	0.098	1.841
`Qwen2-VL-7B`	100.0%	81.0%	0.934	0.934	0.143	1.643	0.894	1.777
`Qwen2.5-7B`	92.9%	88.1%	0.835	0.835	0.649	1.429	0.693	1.791

Table 10: Procedure 1 independent replication for small open-source models. Self: responder model judges its own outputs. GPT: `gpt-5.4-mini` acts as external judge (validated in Table 9)

fragility. At a minimum, benchmark introductions must report run-to-run variance, clearly document all stochastic parameters and seeds, and provide an end-to-end reproduction script independent of complex orchestration layers.

7 CONCLUSION

This paper has attempted to do three things: reproduce the core empirical claims of the Potemkin Understanding paper, characterize the fragilities and failures encountered in that effort, and extend the evaluation to model families and conditions not covered by the original study. Our reproduction confirms the paper's directional finding: standard non-reasoning LLMs do exhibit high incoherence rates under the automated procedure, and the gap between definition accuracy and task application accuracy in the benchmark is real. But the specific numbers are not as stable as the paper's presentation suggests. Run-to-run variance in the automated procedure is large enough to materially affect conclusions for individual models — Potemkin scores vary by as much as 0.31 (31 percentage points) across identical runs, enough to flip whether a model appears above or below chance performance. The procedure fails for smaller models in ways that are not acknowledged and not easily corrected: Qwen2.5-0.5B produced zero valid BBH runs and only 3 of 5 MMLU runs.

We additionally document that the Potemkin rate formula permits values exceeding 1.0, a property unacknowledged in the original work, observed in three of our seven model-benchmark configurations. The benchmark's aggregate potemkin rates inherit unknown measurement error from unvalidated annotation pipelines. And the single most striking finding in the paper, the order-of-magnitude reduction in incoherence for reasoning models, is left unanalysed despite being the result most consequential for how the field should interpret the paper's central claim. We analyse these shortcomings systematically, and propose the steps required to rectify those for the LLM evaluation community.

Acknowledgments

This material is based on work supported by the United States Air Force and Defense Advanced Research Projects Agency (DARPA) under Agreement No. HR0011-24-9-0424. The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of War or the U.S. Government.

References

[1] Usman Anwar, Abulhair Saparov, Javier Rando, Daniel Paleka, Miles Turpin, Peter Hase, {Ekdeep Singh} Lubana, Erik Jenner, Stephen Casper, Oliver Sourbut, {Benjamin L.} Edelman, Zhaowei Zhang, Mario Günther, Anton Korinek, Jose Hernandez-Orallo, Lewis Hammond, Eric Bigelow, Alexander Pan, Lauro Langosco, Tomasz Korbak, Heidi Zhang, Ruiqi Zhong, Seán Héigeartaigh, Gabriel Recchia, Giulio Corsi, Alan Chan, Markus Anderljung, Lilian Edwards, Aleksandar Petrov, {Christian Schroeder} {de Witt}, {Sumeet Ramesh} Motwani, Yoshua Bengio, Danqi Chen, {Philip H.S.} Torr, Samuel Albanie, Tegan Maharaj, Jakob Foerster, Florian Tramer, He He, Atoosa Kasirzade, Yejin Choi, and David Krueger. 2024. Foundational Challenges in Assuring Alignment and Safety of Large Language Models. *Transactions on Machine Learning Research* 2024 (2024). Publisher Copyright: © 2024, Transactions on Machine Learning Research. All rights reserved..

[2] Konstantine Arkoudas. 2023. GPT-4 can't reason. *arXiv preprint arXiv:2308.03762* (2023).

[3] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating Large Language Models Trained on Code. *eprint arXiv: 2107.03374* (2021).

[4] Kenneth Ward Church and Joel Hestness. 2019. A survey of 25 years of evaluation. *Natural Language Engineering* 25, 6 (2019), 753–767. doi:10.1017/S1351324919000275

[5] Mengnan Du, Fengxiang He, Na Zou, Dacheng Tao, and Xia Hu. 2023. Shortcut learning of large language models in natural language understanding. *Commun. ACM* 67, 1 (2023), 110–120.

[6] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300* (2020).

[7] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300* (2020).

[8] Anne Herrmann-Werner, Teresa Festl-Wietek, Friederike Holderried, Lea Herschbach, Jan Griewatz, Ken Masters, Stephan Zipfel, and Moritz Mahling. 2024. Assessing ChatGPT's mastery of Bloom's taxonomy using psychosomatic medicine exam questions: mixed-methods study. *Journal of medical Internet research* 26 (2024), e52113.

[9] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. 2025. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems* 43, 2 (2025), 1–55.

[10] Thomas Huber and Christina Niklaus. 2025. LLMs meet Bloom's Taxonomy: A Cognitive View on Large Language Model Evaluations. In *Proceedings of the 31st International Conference on Computational Linguistics*, Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, and Steven Schockaert (Eds.). Association for Computational Linguistics, Abu Dhabi, UAE, 5211–5246. https://aclanthology.org/2025.coling-main.350/

[11] Mehran Kazemi, Bahare Fatemi, Hritik Bansal, John Palowitch, Chrysovalantis Anastasiou, Sanket Vaibhav Mehta, Lalit K Jain, Virginia Aglietti, Disha Jindal, Yuanzhu Peter Chen, et al. 2025. Big-bench extra hard. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 26473–26501.

[12] David R. Krathwohl. 2002. A Revision of Bloom's Taxonomy: An Overview. *Theory Into Practice* 41, 4 (2002), 212–218.

[13] Walter Laurito, Benjamin Davis, Peli Grietzer, Tomáš Gavenčiak, Ada Böhm, and Jan Kulveit. 2025. AI–AI bias: Large language models favor communications generated by large language models. *Proceedings of the National Academy of Sciences* 122, 31 (2025), e2415697122. arXiv:https://www.pnas.org/doi/pdf/10.1073/pnas.2415697122 doi:10.1073/pnas.2415697122

[14] Dawei Li, Renliang Sun, Yue Huang, Ming Zhong, Bohan Jiang, Jiawei Han, Xiangliang Zhang, Wei Wang, and Huan Liu. 2025. Preference leakage: A contamination problem in llm-as-a-judge. *arXiv preprint arXiv:2502.01534* (2025).

[15] Yiqi Liu, Nafise Sadat Moosavi, and Chenghua Lin. 2024. LLMs as narcissistic evaluators: When ego inflates evaluation scores. In *Findings of the Association for Computational Linguistics: ACL 2024*. 12688–12701.

[16] Marina Mancoridis, Bec Weeks, Keyon Vafa, and Sendhil Mullainathan. 2025. Potemkin Understanding in Large Language Models. In *Forty-second International Conference on Machine Learning*. https://openreview.net/forum?id=oetxkccLoq

[17] Adeladlew Kassie Netere, Anna-Marie Babey, Roisin Kelly-Laubscher, Thomas A. Angelo, and Paul J. White. 2024. Mapping design stages and methodologies for developing STEM concept inventories: a scoping review. *Frontiers in Education* Volume 9 - 2024 (2024). doi:10.3389/feduc.2024.1442833

[18] Arjun Panickssery, Samuel R Bowman, and Shi Feng. 2024. Llm evaluators recognize and favor their own generations. *Advances in Neural Information Processing Systems* 37 (2024), 68772–68802.

[19] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 conference on empirical methods in natural language processing*. 2383–2392.

[20] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Semantically Equivalent Adversarial Rules for Debugging NLP models. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Iryna Gurevych and Yusuke Miyao (Eds.). Association for Computational Linguistics, Melbourne, Australia, 856–865. doi:10.18653/v1/P18-1079

[21] Himadri Nath Saha, Dipanwita Chakraborty Bhattacharya, Sancharita Dutta, Arnab Bera, Srutorshi Basuray, Satyasaran Changdar, Saptarshi Banerjee, and Jon Turdiev. 2026. Transforming Healthcare with State-of-the-Art Medical-LLMs: A Comprehensive Evaluation of Current Advances Using Benchmarking Framework. *Computers, Materials, & Continua* 86, 2 (2026), 1.

[22] David Sands, Mark Parker, Holly Hedgeland, Sally Jordan, and Ross Galloway. 2018. Using concept inventories to measure understanding. *Higher Education Pedagogies* 3, 1 (2018), 173–182.

[23] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on machine learning research* (2023).

[24] Qwen Team. 2024. Qwen2.5 Technical Report. arXiv:2412.15115 [cs.CL] https://arxiv.org/abs/2412.15115

[25] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. *CoRR* abs/1804.07461 (2018). arXiv:1804.07461 http://arxiv.org/abs/1804.07461

[26] Shen Wang, Tianlong Xu, Hang Li, Chaoli Zhang, Joleen Liang, Jiliang Tang, Philip S Yu, and Qingsong Wen. 2026. Large language models for education: A survey and outlook. *IEEE Signal Processing Magazine* 42, 6 (2026), 51–63.

[27] Yaxiong Wu and Yixuan Li. 2026. Agentic FinTech: A Comprehensive Survey on AI Agents in Finance in the Era of LLMs. *Available at SSRN 6136529* (2026).

[28] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems* 36 (2023), 46595–46623.

A Details on Hardware and Software Environment used for reproducibility

Table 11 provides details on the hardware and software environment used in our experiments.

B Research Methods

B.1 Prompt Templates

Concept Detection. "Does the following question test a specific concept? If yes, respond with ANSWER: Yes and CONCEPT: [concept name]. If no, respond with ANSWER: No."

Subquestion Generation. "Write {n} other questions that test whether someone who understands the concepts the question is testing truly understands them. Format each question inside `<question>` `</question>` tags."

Grading. "You are an expert tutor. Determine if the answer is correct or incorrect. Grade it correct only if the answer (including reasoning) is completely correct. End with FINAL ANSWER: followed by either 'correct' or 'incorrect'."

Error Introduction. "Modify the following answer to introduce a subtle error. The error should be subtle but one such that a human who knows the concept would know the answer is incorrect."

C Online Resources

Original paper repository: https://github.com/MarinaMancoridis/PotemkinBenchmark

Component	Specification
Operating System	Ubuntu 24.04.4 LTS
CPU	Intel Core i9-14900K (32 threads)
RAM	64.0 GiB
Storage	2 TB SSD
GPU	2 × NVIDIA Quadro RTX 6000 (48 GB VRAM each)
Python	3.10
Primary libraries	`transformers`, `torch`, `openai`, `anthropic`, `datasets`
Artifact commit	`PotemkinBenchmark` @ commit 39 (ab67a78)
Random seeds	Documented per experiment; default seed=42 for all 5-iteration stability runs

Table 11: Hardware and software environment used for reproduction and extension experiments