

The Benchmark Is Not the Construct: Toward Distribution-Based Evaluation in NLP

Haithem Afli

Department of Computer Science
Munster Technological University
Haithem.Afli@mtu.ie

Manuel Frank

Department of Computer Science
Munster Technological University
Manuel.Frank@zohomail.eu

Abstract

Current NLP evaluation often treats a scalar score on a fixed benchmark dataset as if it directly measured the construct the benchmark is intended to represent. This *Measurement-as-Instance* paradigm is appropriate for a fixed evaluation configuration, but can suppress uncertainty, variance, and robustness when a single result is used to support broader capability or generalisation claims. In this position paper, we argue for extending static evaluation with *Measurement-as-Distribution*: a structured evaluation procedure involving principled data generation, validated perturbations, repeated evaluation, and distributional reporting. The framework is configurable: the amount of distributional evidence required should scale with the strength and breadth of the claims being made. We synthesise existing robustness, dynamic-evaluation, prompt-sensitivity, and uncertainty-reporting practices into a general measurement-theoretic framework for distribution-based NLP evaluation and provide a call to action for more rigorous reporting. Using the protocol $\mathcal{P} = (\mathcal{C}, G, \Phi, E, R)$, we identify three risks of relying exclusively on fixed scalar evaluation and derive implications for peer review, benchmark publication, leaderboard reporting, and computational cost. Two case studies demonstrate feasibility: a SQuAD evaluation of 14 models reveals modest but axis-specific performance changes and stable overall rankings with uncertainty-aware comparisons, while a verifier-backed Knights-and-Knaves study illustrates how perturbations of rigid logical tasks can be generated and validated formally.

1 Introduction

The NLP community has produced hundreds of benchmarks over the past decade. For generative large language models (LLMs), benchmarks such as GLUE (Wang et al., 2018), MMLU (Hendrycks et al., 2021), GSM8K (Cobbe et al., 2021), and

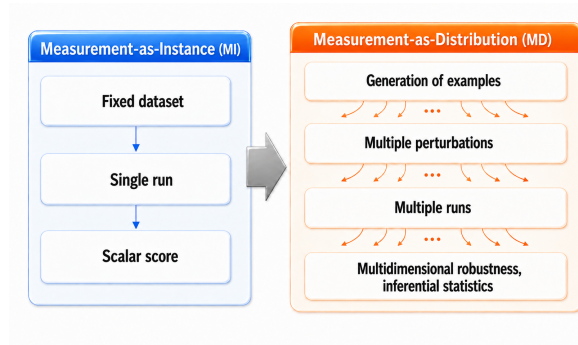


Figure 1: **From instance-based to distribution-based evaluation.** *Left:* Measurement-as-Instance evaluates a fixed dataset once and reports a single scalar score. *Right:* Measurement-as-Distribution generates examples, applies multiple perturbations, repeats evaluation across runs, and reports a multidimensional robustness profile with inferential statistics (simplified, details in Figure 2).

BIG-bench (Srivastava et al., 2023), together with leaderboards such as the Artificial Analysis Intelligence Index¹, provide shared references for comparing systems and reporting progress. For embedding models, MTEB (Muennighoff et al., 2023) and MMTEB (Enevoldsen et al., 2025) play a similar role.

Each benchmark operationalises an abstract capability as a fixed collection of items with ground-truth responses and produces a scalar score. We call this *Measurement-as-Instance* (MI). MI is appropriate when the claim concerns performance under that particular evaluation configuration. However, a fixed evaluation provides limited evidence about how performance varies across other valid evaluation instances or measurement conditions. The issue is therefore not that fixed benchmarks are invalid, but that a single configuration supports narrower claims than those about robustness, generalisation, or performance across broader conditions.

¹<https://artificialanalysis.ai/>

An important distinction is between uncertainty *within a fixed evaluation* and variation *across valid evaluation conditions*. A benchmark can quantify sampling uncertainty over its fixed items, for example through bootstrap confidence intervals, without testing whether the estimate changes when the same construct is instantiated through other valid items, presentations, or conditions. MI therefore estimates performance conditional on a particular configuration, whereas broader claims require evidence about variation across relevant configurations.

Three developments make this distinction increasingly consequential. *Scale and training breadth*: models are trained on corpora derived from the same web sources as evaluation datasets, making statistical overlap difficult to avoid (Chen et al., 2025; Zhou et al., 2023). *Optimisation pressure*: published benchmark scores encourage techniques that perform well on existing tests, potentially increasing the divergence between benchmark performance and construct-level capability (Mirzadeh et al., 2025). *Saturation*: as models improve, scores cluster near the ceiling, reducing discriminative power when models are increasingly similar (Kiela et al., 2021; Akhtar et al., 2026; Ott et al., 2022; Qian et al., 2026; Zhou et al., 2026).

AI benchmarks can be viewed as analogous to psychometric measurement (Blili-Hamelin and Hancox-Li, 2023; Freiesleben and Zezulka, 2025). The relevant analogy is not that fixed tests are invalid, but that repeated use of the same test and reporting a single score do not reveal how performance varies across other valid measurement conditions.

Related work on construct validity similarly argues that benchmark scores are evidence about a target phenomenon only insofar as the task, metric, and resulting claims validly operationalise that phenomenon (Bean et al., 2025; Kearns, 2026; Raji et al., 2021). We build on this view but focus on the evaluation procedure itself: even a well-motivated benchmark can yield overconfident conclusions when treated as a single fixed instance rather than as one draw from a broader space of valid evaluation instances.

The central argument is: **benchmarks measure how a model performs under a particular evaluation configuration, while broader claims require evidence about how that performance varies across a distribution of valid evaluation conditions**. Distribution-based evaluation does not

imply that fixed benchmark scores are inadequate for every purpose, nor does it require rankings to change. Rather, it provides richer evidence about robustness, uncertainty, and differences between models that may achieve similar static scores but behave differently under valid variation.

We formalise this extension as an evaluation protocol, $\mathcal{P} = (\mathcal{C}, G, \Phi, E, R)$, covering the evaluation construct, data-generation process, perturbations, evaluation procedure, and reporting strategy. The protocol is configurable: the amount of distributional evidence can be adapted to the strength and breadth of the claim. Measurement-as-Distribution therefore complements rather than replaces existing static benchmarks and robustness-oriented evaluation frameworks.

Our contribution is not any individual component of this protocol. Data generation, perturbation testing, prompt-sensitivity analysis, repeated evaluation, uncertainty reporting, and rank-stability analysis all exist in prior work (see Section 4). Our contribution is to organise these components into a single configurable measurement protocol that is not restricted to generative or embedding models. Relevant robustness work includes evaluations of generative models (Mirzadeh et al., 2025; Mizrahi et al., 2024; Sclar et al., 2024; Zhang et al., 2026; Zhu et al., 2024a,b) and embedding models (Frank and Afli, 2026b; Goel et al., 2025; Kostiuik and Enevoldsen, 2026; Frank and Afli, 2026a).

We demonstrate feasibility through two complementary case studies. A SQuAD evaluation of 14 models applies four perturbation axes and repeated evaluation to examine robustness profiles, rank stability, and uncertainty. A verifier-backed Knights-and-Knaves case study shows how the same framework can be applied to a rigid logical task whose generated instances and perturbations can be checked against formal validity constraints. These examples illustrate that distribution-based evaluation can provide additional evidence without replacing established benchmark tasks.

We first contrast *Measurement-as-Instance* with *Measurement-as-Distribution*, then identify failure modes of fixed scalar evaluation, formalise the protocol, relate it to existing work, state testable predictions, and examine the two case studies before deriving implications for peer review, benchmarks, leaderboards, and computational cost.

2 Two Paradigms: Instance vs. Distribution

Measurement-as-Instance (MI). A benchmark $\mathcal{B}$ under this paradigm consists of an explicit construct specification and a fixed collection of evaluation datasets, each paired with a labelling function and a metric,

$$\mathcal{B} = (\mathcal{C}, \{(\mathcal{D}_k, \mathcal{L}_k, \mathcal{M}_k)\}_{k=1}^K),$$

where $\mathcal{C}$ specifies the intended construct, $\mathcal{D}_k$ is a dataset of items, $\mathcal{L}_k$ assigns correct outputs, and $\mathcal{M}_k$ scores model outputs on dataset k . Evaluation produces dataset-level scores $\mathcal{M}_k(f, \mathcal{D}_k, \mathcal{L}_k)$ or an aggregate. The defining feature is a fixed configuration of datasets, labels, metrics, prompts, and aggregation choices. A static benchmark may therefore specify its construct, validate its items, and quantify uncertainty conditional on its fixed sample; what MI does not by itself measure is how results vary across other valid evaluation configurations.

Measurement-as-Distribution (MD). MD extends MI by introducing the protocol

$$\mathcal{P} = (\mathcal{C}, G, \Phi, E, R),$$

where $\mathcal{C}$ is an explicit construct specification; G is a data-generation function; $\Phi = \{\phi_1, \dots, \phi_k\}$ is a validated family of perturbations that preserve the target construct and task structure while varying admissible measurement conditions; E is a multi-run evaluation procedure; and R produces a robustness profile including distributional summaries, confidence intervals, and stability statistics.

The distinction is not whether uncertainty can be estimated, but which sources of variation are represented in the measurement process. MI conditions inference on one fixed configuration, whereas MD makes variation across generated instances, validated perturbations, and repeated evaluation conditions explicit. A model that performs consistently across these conditions provides evidence about robustness that cannot be obtained from its score on the original configuration alone. MD therefore complements rather than replaces static benchmark scores. The required amount of distributional evidence can be scaled to the claim being made.

Connection to measurement theory. This reframing has a formal precedent in Generalizability Theory (G-theory; Cronbach et al., 1972), which

treats a performance score as an estimate of a universe score and characterises reliability through variance components attributable to items, occasions, and measurement conditions. A fixed benchmark conditions on one configuration of these factors, whereas systematic sampling across relevant sources of variation can make their contribution to observed performance explicit. MD extends this measurement perspective from performance under a fixed configuration to variation across a declared space of valid conditions.

3 Failure Modes of Instance-Based Measurement

The failure modes below arise when a fixed evaluation configuration is used to support claims extending beyond that configuration. They are not inherent flaws of static benchmarks: MI can specify constructs, validate items, and quantify uncertainty conditional on its fixed sample. The limitation is that variation outside that configuration remains unmeasured unless explicitly incorporated.

Benchmark leakage: the fixed instance becomes a target. When a model is trained on data statistically correlated with $\mathcal{D}$, the evaluation is no longer independent of the measured object. Zhou et al. (2023) demonstrate that leakage inflates scores without corresponding capability gains, while Zhang et al. (2024) report up to 8 percentage points higher accuracy on contaminated GSM8K than on matched GSM1k. Balunovic et al. (2025) confirm the effect at scale, finding substantially lower scores on competition mathematics evaluated at problem release than on previously fixed sets. A fixed score alone cannot diagnose benchmark-specific adaptation; under MD, gaps between fixed-benchmark performance and fresh draws from G or validated variants from Φ provide diagnostic evidence.

Prompt sensitivity: the scalar suppresses prompt variance. Across 6.5 million evaluation instances, 20 models, and 39 tasks, Mizrahi et al. (2024) show that instruction templates can substantially change performance estimates and relative rankings. Kostiuk and Enevoldsen (2026) similarly find prompt sensitivity in instruction-tuned embedding models. Prompt choice therefore constitutes an additional source of measurement variation that is hidden when a single template is fixed. Under MD, prompt sensitivity can be reported as

an explicit auxiliary analysis, separately from the primary perturbation axes.

Community-level overfitting: the score-construct relation degrades. Because the relationship between score and construct is established at benchmark construction, its validity may not be reassessed as models and evaluation practices evolve. [Mirzadeh et al. \(2025\)](#) show that adding seemingly relevant but irrelevant information to GSM8K-style problems can reduce accuracy by up to 65 percentage points. [Wang et al. \(2024\)](#) demonstrate that LLM-judge rankings can reverse under response-order manipulation, while [Koo et al. \(2024\)](#) report biased comparisons in 40% of cases and an average human-model Rank-Biased Overlap of 44%. Under MD, the explicit construct specification $\mathcal{C}$ provides a reference against which construct drift and discriminability can be monitored, for example using the Discriminability Score of [Qian et al. \(2026\)](#).

4 A Formal Evaluation Protocol

In this section, we formalise MD as a protocol for specifying the target construct, valid evaluation variation, repeated evaluations, and distributional reporting. [Figure 2](#) visualises the protocol. A more explicit algorithmic specification is provided in [Appendix A](#). The specification for a concrete example is provided in [Appendix B](#).

We define distribution-based evaluation as a protocol

$$\mathcal{P} = (\mathcal{C}, G, \Phi, E, R),$$

where the components specify the target construct, the valid data-generation process, the family of construct-preserving transformations, the evaluation procedure, and the reporting obligations.

Construct specification. $\mathcal{C}$ defines the target capability and separates construct-relevant from construct-irrelevant variation. A construct specification should state what the evaluation is intended to measure, what kinds of examples are acceptable, which features must be preserved, and which may vary. It should also connect the intended claims with the experimental design. This specification provides the reference against which generated instances and perturbations are validated and against which the scope of the resulting claims is assessed.

Data generation. G samples or generates valid evaluation instances conditioned on $\mathcal{C}$. The generator may resample from an existing benchmark, select datasets according to stated criteria, generate new instances from templates or models, generate completely new datasets, or combine these choices. For a fixed benchmark, G may simply return the same predefined evaluation set; MD makes the choice of sampling or generation explicit when variation across evaluation instances is part of the claim.

Perturbation procedures. $\Phi = \{\phi_1, \dots, \phi_k\}$ defines construct-preserving transformations that vary admissible measurement conditions while preserving the intended label or target relation. Because preservation is construct-dependent, each perturbation procedure should be validated. As a minimal requirement, the protocol should provide evidence for the construct-validity of each perturbation. In many studies, this will include human validation; for rigid tasks with executable specifications, formal or verifier-based validation may instead provide the relevant evidence. Validating a subsample is often more practical because it controls annotation cost, but it does not validate every generated variant. Runtime automatic validation can therefore complement human or verifier-based validation.

The protocol should specify the validation design, acceptance criteria, runtime checks, fallback rules, and automatically detected failure rates. In particular, failed perturbations should be recorded explicitly rather than silently treated as valid variants. Additionally, inter-rater agreement should be measured when human annotation is used; see [James \(2026\)](#) for a recent overview of metrics. We use $\alpha : \Phi \rightarrow \mathcal{A}$ to map a perturbation ϕ_j to a robustness axis $\alpha(\phi_j) \in \mathcal{A}$. Robustness axes may be differently motivated, e.g. linguistically. However, by default, we suggest using deployment-relevant robustness axes.

Evaluation procedure. E specifies how models are evaluated based on generated and perturbed evaluation instances. It defines the sample size N , the number of runs or seeds N_{runs} , the seeds $\mathcal{S} = \{s_1, \dots, s_{N_{\text{runs}}}\}$, the scoring procedure, and the records retained for downstream analysis. In run r , the seed s_r is used for sampling and, where relevant, for stochastic components of evaluation such as perturbation generation and either answer or embedding generation. The protocol should

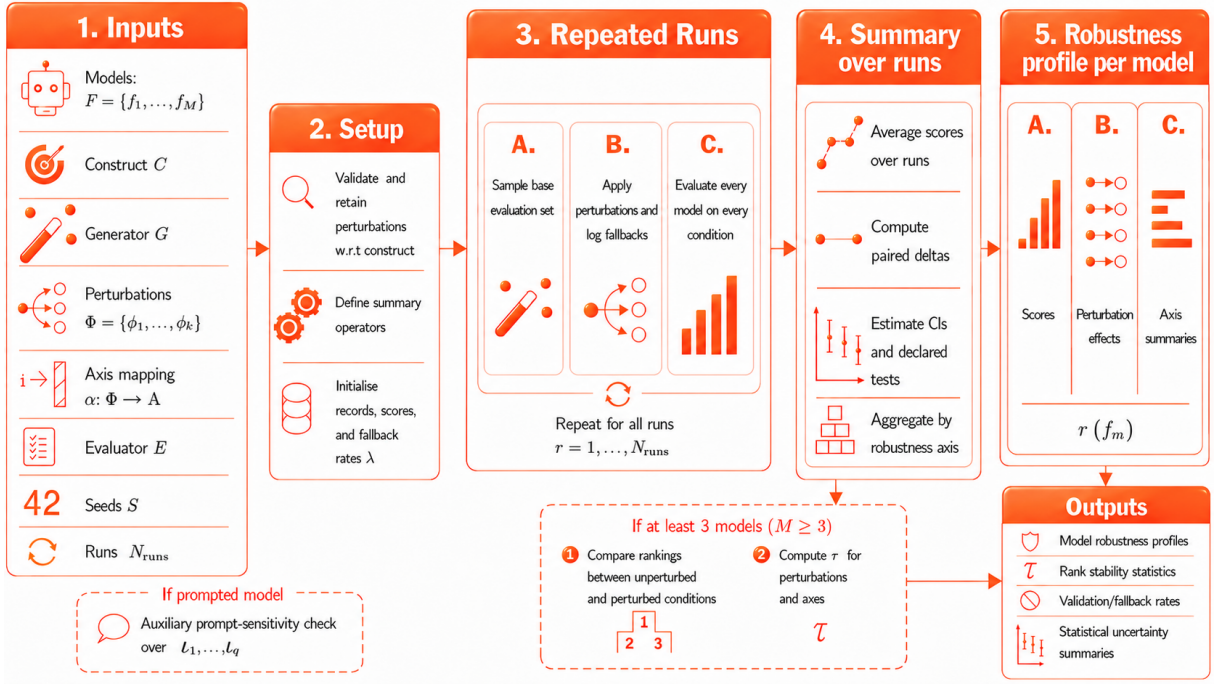


Figure 2: Overview of the Measurement-as-Distribution evaluation protocol. Across repeated runs, the protocol samples evaluation sets, generates validated perturbation variants, evaluates models on all variants, and summarises results into robustness profiles with uncertainty and rank-stability statistics. For prompted models, prompt sensitivity is treated as a separate auxiliary analysis. Formal specification in [Appendix A](#).

also specify which factors are resampled across runs and which are held fixed. For example, when several observations belong to the same higher-level unit, such as multiple questions from the same passage, that unit may be held fixed across seeds while stochastic components such as perturbation generation or model inference vary.

For prompted models, the main protocol uses a primary instruction ι^* to keep evaluation tractable. Unless the claim is specifically about instruction following, prompt sensitivity is assessed separately through an auxiliary analysis, for example by reporting scores for at least three semantically equivalent prompts. This keeps prompt sensitivity explicit without conflating it with the primary perturbation axes.

Reporting function. R maps the resulting evaluations to robustness profiles rather than a single scalar score. At minimum, these profiles should include original and perturbed scores, paired deltas relative to the unperturbed condition, uncertainty intervals, perturbation validation outcomes such as acceptance or fallback rates, and explicit p -values where hypothesis tests are used. When multiple hypothesis tests are reported, p -values should be corrected for multiple comparisons within the de-

clared test family. We recommend Holm correction as a default because, like Bonferroni correction, it controls the family-wise error rate while being less conservative than the Bonferroni procedure ([Holm, 1979](#)).

For model comparisons, the declared test family should reflect the claims being made. For example, when introducing a new or improved model, we suggest testing the paired score difference under the unperturbed condition and under each perturbation axis compared to a declared reference model such as the currently leading model. However, perturbation-level and axis-level effects can also be reported descriptively without formal hypothesis tests.

Rank-stability statistics using Kendall’s τ should be reported when at least three models are compared. The primary rank-stability analysis should compare the unperturbed ranking with each axis-specific perturbed ranking separately, rather than only with a single ranking aggregated across robustness axes.

R should also specify the unit of analysis used for uncertainty estimation and hypothesis testing. When observations are paired or share a higher-level unit, the analysis should preserve these dependencies rather than treat all observations as indepen-

dent. This is particularly important when multiple questions are sampled from the same passage or when multiple perturbations are derived from the same parent instance.

The resulting protocol therefore makes explicit which aspects of the evaluation are fixed and which are sampled or varied. A static benchmark can be recovered as a restricted configuration in which the data-generation process returns a predefined dataset and no perturbation variation is introduced. MD is useful when additional variation is relevant to the claim and can be specified and validated without changing the underlying construct.

Recent work increasingly treats benchmarking not only as a scoring problem, but as a measurement problem concerning a target phenomenon. [Bean et al. \(2025\)](#) review 445 benchmark papers and argue that benchmark design should define the target phenomenon, justify the task and metric, report uncertainty, address contamination, and establish that benchmark scores support the claims made. [Kearns \(2026\)](#) similarly model benchmark scores as imperfect indicators of latent LLM capabilities, while recent work on the “proxy presumption” shows how embedding-based measures can conflate target constructs with nuisance factors such as topic, style, and authorship ([Li et al., 2026](#)). [Mondal and Bhardwaj \(2026\)](#) extend this concern to deployment, arguing that benchmark scores may fail to capture stakeholder-centred utility.

This work supports our starting point: benchmark scores are informative only insofar as the benchmark validly operationalises the phenomenon it claims to measure. These accounts primarily address the validity of mappings from scores or benchmark outcomes to target constructs. Our contribution is complementary: we focus on the evaluation procedure itself. Construct validity specifies what a benchmark is intended to represent; MD specifies how relevant sources of variation can be generated, validated, repeated, and reported. Broader benchmark-based capability claims therefore require evidence beyond a single fixed evaluation configuration.

Dynamic Evaluation Frameworks. The empirical motivation for MD comes from work that treats benchmark evaluation as a multidimensional procedure rather than a single fixed score. [Chen et al. \(2025\)](#) survey the shift from static to dynamic benchmarking for generative models. Embedding evaluation provides particularly clear examples of

this broader idea: existing instances can be paraphrased or otherwise transformed while preserving task semantics, with resulting score variation analysed across models, languages, and tasks.

MTEB ([Muennighoff et al., 2023](#)) and MMTEB ([Enevoldsen et al., 2025](#)) standardised broad embedding evaluation across diverse tasks. PTEB ([Frank and Afi, 2026b](#)) introduced stochastic paraphrase-based evaluation across 20 datasets and 25 languages, while HTEB ([Frank and Afi, 2026a](#)) extended this approach to multiple perturbation axes, human validation, repeated evaluation, and multidimensional robustness profiles. HTEB is the closest empirical precursor to MD because it makes input variation an explicit component of evaluation; however, its scope is narrower, focusing on embedding models and input perturbation.

Other frameworks vary different parts of the evaluation process. Dynabench ([Kiela et al., 2021](#)) uses human- and model-in-the-loop adversarial data collection, while Dynaboard ([Ma et al., 2021](#)) varies evaluation and reporting criteria through multi-metric leaderboards. TextFlint ([Wang et al., 2021](#)), CheckList ([Ribeiro et al., 2020](#)), Robustness Gym ([Goel et al., 2021](#)), and HELM ([Liang et al., 2023](#)) provide complementary approaches to robustness, behavioural testing, subpopulation analysis, perturbation, and multi-metric evaluation. SAGE ([Goel et al., 2025](#)) further evaluates embedding models across transformation robustness, information sensitivity, clustering, retrieval, and human-preference alignment. Counterfactual prompting also highlights the need to interpret perturbation effects relative to appropriate baselines ([Yang et al., 2026](#)).

Across these approaches, the relevant distinction is which part of the measurement process is varied and how that variation is validated and reported. MD makes this abstraction explicit by separating construct specification, data generation, perturbations, repeated evaluation, and distributional reporting in a model- and task-agnostic framework. Its contribution is therefore not an individual robustness or dynamic-evaluation technique, but the organisation of existing components into a configurable measurement protocol whose components can be selected according to the claim being made.

Statistical Analysis for NLP. Previous work has established that single evaluations can be unreliable and that score distributions should instead be analysed ([Reimers and Gurevych, 2018](#); [Dror et al.,](#)

2019). For generative LLM benchmarks, [Zhang et al. \(2026\)](#) similarly show that relying on a single sampled generation can produce unstable estimates and ranking errors. More broadly, benchmark task selection and aggregation can affect model rankings ([Dehghani et al., 2021](#)), while live human-preference leaderboards can be affected by platform-specific evaluation policies ([Singh et al., 2025](#)). NLP research has also established inferential methods and the need for correction for multiple comparisons ([Evert, 2004](#); [Søgaard, 2013](#); [Søgaard et al., 2014](#); [Yeh, 2000](#); [Dror et al., 2018, 2017](#)), with recent work revisiting broader evaluation concerns ([Dhar and Søgaard, 2026](#)). MD builds on these practices by integrating uncertainty, paired effects, rank stability, and validation outcomes with dynamic and robustness-oriented evaluation.

Prompt Sensitivity. Prompt choice introduces an additional source of measurement variation. Instruction paraphrases can change both model scores and relative rankings ([Mizrahi et al., 2024](#)), while meaning-preserving formatting changes and small perturbations such as typos can also affect performance ([Sclar et al., 2024](#); [Zhu et al., 2024a](#)). Prompt sensitivity has also been observed in instruction embedding models ([Kostiuk and Enevoldsen, 2026](#)). We therefore treat prompt choice as an explicit evaluation factor, reported separately unless the claim specifically concerns instruction following, avoiding conflation with other construct-preserving perturbation axes.

5 Testable Predictions

The protocol yields three empirical commitments for distribution-based evaluations. The predictions are motivated by the failure modes in [Section 3](#). Our case studies provide initial evidence for these commitments, while their broader generality remains an empirical question.

P1: Rank-score dissociation. Distribution-based evaluation may reveal differences between unperturbed and perturbed rankings, but ranking stability alone does not establish robustness equivalence. Evaluations should therefore report rank stability, for example with Kendall’s τ , together with axis-specific score deltas. We expect both outcomes to be possible: perturbations may change rankings, while highly stable rankings may still conceal non-trivial differences in performance tra-

jectories. *Ranking stability does not imply robustness equivalence*: models with identical rankings may respond differently across perturbation axes.

P2: Score variation under valid perturbation. For construct-preserving perturbations, performance may vary relative to the unperturbed condition, but the direction and magnitude need not be universal. The mean paired delta, $\Delta = \text{SCORE}_{\text{perturbed}} - \text{SCORE}_{\text{unperturbed}}$, should therefore be reported for each perturbation or robustness axis together with uncertainty estimates. Negative deltas indicate degradation, while near-zero or positive deltas may indicate robustness or task-specific sensitivity. Following PTEB and HTEB, the protocol treats the direction, magnitude, and uncertainty of Δ as empirical quantities rather than assuming a uniformly negative effect or statistical significance.

P3: Robustness multidimensionality. Models may exhibit dissociated robustness profiles across perturbation axes, such that per-axis deltas provide information lost in an aggregate score. Motivated by HTEB and examined in our SQuAD case study, this hypothesis should be evaluated through per-axis delta estimates, uncertainty intervals, inter-axis relationships, and profile-level comparisons. Even stable overall rankings may conceal meaningful differences in responses to valid evaluation variation.

6 Case Studies

We demonstrate MD through two complementary case studies: SQuAD ([Rajpurkar et al., 2016](#)), representing a well-established QA benchmark with linguistic and presentation variation, and Knights & Knaves (K&K), a rigid logical task whose generated instances can be validated by an executable verifier. Experimental details and validation procedures are provided in [Appendix B](#) and [Appendix C](#).

6.1 SQuAD

We evaluated 14 models on 3,000 QA items sampled from 1,000 paragraphs of SQuAD v1.1 using *G*. Four perturbation axes were evaluated: (1) question rephrasing, (2) additional context, (3) list-structured passages, and (4) typos. Each perturbation was validated using its runtime acceptance criteria, with $N_{\text{runs}} = 6$.

Perturbations produced lower average scores than the unperturbed evaluation, although the mag-

nitude varied across models and axes. Rankings remained highly stable, with Kendall’s $\tau \in [0.956, 1.000]$, but some relative positions changed. For example, RoBERTa-base and XLM-R-large score 91.2% and 91.1%, respectively, on the original evaluation, but 90.6% and 90.7% averaged across perturbations, reversing their ordering. Thus, high rank stability does not imply equivalent robustness trajectories.

Validation outcomes were 100% for question rephrasing and presentation structure, 99.93% for surface noise, and 82.94% for context composition, corresponding to a 17.06% fallback rate for the latter. Rejected variants were explicitly excluded rather than treated as valid evaluations. For the predefined RoBERTa-base versus XLM-R-large comparison, all bootstrap confidence intervals included zero and the corresponding p -values were not significant before multiplicity correction. The ranking reversal should therefore be interpreted as an illustration of distributional variation rather than evidence of a statistically reliable model difference.

As a generator-sensitivity check, we repeated perturbation generation with Qwen-3.8-27B instead of Gemma-4-31B for $N_{\text{runs}} = 3$. Both generators produced the same model ranking across the evaluated conditions. This check treats generator choice as a potential source of measurement variation.

6.2 Knights & Knaves

We use the Knights & Knaves task of Xie et al. (2025) as a verifier-backed feasibility study. The task requires determining unique Knight/Knave assignments under explicit truthfulness constraints; the original problem formulation is given in Appendix C.

Applying MD, G generated 100 valid parent instances with respect to $\mathcal{C}$, with an acceptance rate of 28.2%. For each accepted parent, E applied Φ five times, with each perturbation changing exactly one literal. Perturbed instances were independently checked by the verifier and retained only when they satisfied the predefined validity criteria, giving a perturbation acceptance rate of 12.5%. We evaluated Gemma-4-31B and Qwen-3.8-27B; the resulting error profiles are reported in Table 1. The purpose of this case study is to demonstrate that MD can be applied to rigid symbolic tasks using formal validation, rather than to provide a statistical comparison between the two models.

Model	Assignment	Formatting	Total
Gemma-4	15	0	15 (2.5%)
Qwen-3.8	6	5	11 (1.8%)

Table 1: K&K role-assignment and unrecoverable formatting errors over 100 generated parent instances and their 500 attempted perturbed instances.

7 Implications for ACL and the Field

The field already has the tools needed for richer evaluation: principled data generation, perturbation-based robustness checks, prompt-sensitivity analyses, repeated evaluations, and uncertainty quantification. What is missing is more consistent use of these tools in proportion to the claims being made. A scalar score on a fixed benchmark can be appropriate for a narrow claim about that configuration, whereas claims about robustness, generalisation, or broader progress require evidence across relevant evaluation conditions. Our recommendations are therefore pragmatic: they do not require replacing existing benchmarks or applying the full MD protocol in every study. Instead, the amount of distributional evidence should scale with the strength and breadth of the claim.

Peer review standards. Static scalar benchmark improvements should not be treated as sufficient evidence for broad claims of progress on their own. Reviewers should consider four practical requirements: (i) an *explicit data-generation or dataset-selection protocol* that clarifies how evaluation data were obtained and the scope of the resulting claim; (ii) *perturbation-sensitivity reporting* for claims about robustness or generalisation, using at least one construct-preserving perturbation where appropriate; (iii) *uncertainty-aware reporting*, including confidence intervals and appropriate correction for multiple comparisons when hypothesis tests are used; and (iv), for instruction-following models, *multi-prompt reporting* (Mizrahi et al., 2024) when results may depend on instruction choice. If a proposed analysis is inappropriate for a task, authors should explain why. These requirements establish a proportional reporting floor rather than mandating full MD adoption.

Benchmark publication norms. New benchmarks should specify the target construct $\mathcal{C}$, identify relevant construct-preserving variation where applicable, document at least one validated variation or perturbation procedure and its rejection

criteria, and report discriminability at the current state of the art (Qian et al., 2026). Bowman and Dahl (2021) identified the need for stronger benchmark design practices, but these have not become systematic publication requirements. The required level of variation and validation should again depend on whether the benchmark is intended to support fixed-instance claims or broader capability and robustness claims.

Leaderboard reporting. Leaderboards should report aggregate scores together with uncertainty estimates and, where robustness or generalisation is part of the intended claim, per-axis robustness information. They should also indicate when a benchmark approaches saturation and no longer sufficiently discriminates among current models. If perturbation-based evaluation is inappropriate, this should be stated explicitly. This would distinguish a stable ordering on one fixed benchmark from evidence that model differences persist across relevant evaluation conditions.

Reporting of statistical methods. Authors should clearly specify the statistical methods used for evaluation, including the estimation or testing procedure, unit of analysis, number of observations, and treatment of multiple comparisons. Confidence intervals and p -values should be accompanied by sufficient methodological detail to reproduce the analysis and a brief justification of the chosen method. An appendix table summarising these choices can provide a concise reporting format (see Appendix B and Table 6).

Computational cost. MD increases evaluation cost through additional sampling, perturbation generation, validation, and repeated runs. This overhead should be viewed as part of supporting stronger claims rather than as a requirement for every evaluation. The protocol is intentionally configurable: researchers can adjust the number of instances, perturbation axes, validation effort, and repeated runs to match the claim. A minimal distributional evaluation, such as one validated perturbation axis with uncertainty intervals, can already provide additional evidence, while stronger claims about robustness, generalisation, or state-of-the-art performance warrant broader sampling, multiple relevant axes, and more extensive validation.

8 Conclusion

We have argued that *Measurement-as-Instance*, which evaluates a fixed benchmark configuration and reports performance for that configuration, provides useful but necessarily scoped evidence. *Measurement-as-Distribution*, formalised as $\mathcal{P} = (\mathcal{C}, G, \Phi, E, R)$, extends this evaluation by making relevant sources of variation explicit through generated instances, validated perturbations, repeated runs, and distributional reporting. The protocol can be configured according to the claims being made: the stronger or broader the claim, the more distributional evidence is required.

The central claim is not that distribution-based evaluation will always change rankings or reveal large effect sizes. Rather, it makes uncertainty, robustness variation, and benchmark–construct alignment more empirically inspectable. Our case studies provide initial evidence that this additional information can be obtained without replacing established benchmark tasks: the SQuAD study reveals axis-specific performance variation and highly stable but not identical rankings, while the Knights & Knaves study demonstrates that verifier-backed generation and perturbation can support distributional evaluation of a rigid logical task. These examples do not establish that the observed patterns generalise to all NLP tasks or model families. They instead illustrate the central methodological argument: a fixed benchmark score is one measurement under a particular configuration, whereas broader claims require evidence about how performance behaves across relevant valid configurations. The field already has many of the concepts and tools needed to implement *Measurement-as-Distribution*; what remains is to make their use proportional to the claims being made and more consistently reported.

Limitations

This paper is a position paper and does not introduce a new benchmark or a large-scale comparison of models. Its contribution is conceptual and methodological: it formalises distribution-based evaluation as a general protocol and derives reporting recommendations from existing work on robustness, dynamic evaluation, and statistical uncertainty. The case studies in Section 6 provide initial feasibility evidence and illustrate how the protocol can be instantiated in two different settings, but they are not intended to establish the generality of

the framework across tasks, datasets, or model families. The predictions in [Section 5](#) should therefore be understood as empirical commitments whose broader validity remains to be tested across a wider range of evaluation settings.

A central limitation of MD is that perturbation validity is construct-dependent. A transformation that preserves the target construct in one setting may change it in another. For example, paraphrasing may preserve semantic similarity for some sentence-pair tasks but alter pragmatic meaning, factual specificity, or discourse structure in other tasks. The protocol therefore does not remove the need for construct validation. Instead, it makes this requirement explicit: perturbation procedures should be justified, validated, and reported with rejection rates or other evidence that the intended relation has been preserved. For tasks with executable specifications, formal or verifier-based validation can provide an alternative to human validation, as illustrated by the Knights & Knaves case study.

The protocol is also limited by the quality and biases of the mechanisms used to generate evaluation variation. When perturbations are produced by LLMs, they may introduce artefacts, simplify inputs, alter style in systematic ways, or amplify demographic and cultural biases. These effects can make robustness estimates depend partly on the generator rather than only on the evaluated model. Our generator sensitivity analysis in the SQuAD case study provides an initial check of this issue, but it is limited to the generators and experimental conditions considered there. Distribution-based evaluation should therefore not be interpreted as automatically more valid than static evaluation; its validity depends on the design and validation of the sampled instances and perturbations.

Finally, distribution-based evaluation does not eliminate benchmark gaming or contamination. If distributional protocols become standardised, they too may become targets of optimisation. The advantage of the proposed framework is not that it makes evaluation immune to overfitting, but that it exposes more of the measurement process and provides additional diagnostic information when benchmark–construct alignment deteriorates. Maintaining that advantage will require periodically revising generators, perturbation families, validation procedures, and reporting norms.

Ethical Considerations

Barriers for resource-constrained research groups. As discussed in [Section 7](#), MD can increase the cost of evaluation. Repeated runs, perturbation generation, and human validation increase compute requirements and annotation cost. This may disproportionately affect smaller research groups, as well as researchers working in resource-constrained settings. For this reason, we do not argue that every paper must implement all components of the protocol. Instead, we derive pragmatic, claim-dependent recommendations in [Section 7](#).

At the same time, distributional evaluation does not only consist of cost-increasing components. More rigorous statistical reporting, uncertainty estimation, and explicit description of evaluation configurations can often be implemented without substantially increasing computational cost. These components provide ways for resource-constrained researchers to strengthen their evaluation even when perturbation generation or human validation must remain limited in scope.

Bias amplification. LLM-based perturbation generators may amplify social biases present in training data. Perturbation outputs should therefore be audited for demographic and other systematic biases before being used in evaluation contexts. Generator choice and validation procedures should be documented so that observed robustness differences can be distinguished, where possible, from artefacts introduced by the generation process.

Environmental cost. Repeated evaluation, perturbation generation, and model inference can incur higher energy costs than a single static evaluation. Researchers should therefore consider the computational cost of the chosen protocol when selecting the number of runs, instances, and perturbation axes. Because MD is configurable, evaluation effort can be scaled to the strength of the claim rather than maximised by default.

AI usage statement. AI was used to assist with writing, including language polishing and figure preparation. AI was also used to assist with the implementation of the case studies. The authors reviewed all AI-assisted changes and take full responsibility for the final content.

Acknowledgments

We thank the anonymous reviewers and the Area Chair for their constructive feedback, which helped improve the paper.

This research was partially supported by the Horizon Europe project GenDAI (Grant Agreement ID: 101182801) and by the ADAPT Research Centre at Munster Technological University. ADAPT is funded by Taighde Éireann – Research Ireland through the Research Centres Programme and co-funded under the European Regional Development Fund (ERDF) via Grant 13/RC/2106_P2.

References

- Mubashara Akhtar, Anka Reuel, Prajna Soni, Sanchit Ahuja, Pawan Sasanka Ammanamanchi, Ruchit Rawal, Vilém Zouhar, Srishti Yadav, Chenxi Whitehouse, Dayeon Ki, Jennifer Mickel, Leshem Choshen, Marek Šuppa, Jan Batzner, Jenny Chim, Jeba Sania, Yanan Long, Hossein A. Rahmani, Christina Knight, and 18 others. 2026. When AI Benchmarks Plateau: A Systematic Study of Benchmark Saturation. <https://arxiv.org/abs/2602.16763v1>.
- Mislav Balunovic, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. 2025. [MathArena: Evaluating LLMs on uncontaminated math competitions](#). In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.
- Andrew M. Bean, Ryan Othniel Kearns, Angelika Romanou, Franziska Sofia Hafner, Harry Mayne, Jan Batzner, Negar Foroutan, Chris Schmitz, Karolina Korgul, Hunar Batra, Oishi Deb, Emma Beharry, Cornelius Emde, Thomas Foster, Anna Gausen, María Grandury, Simeng Han, Valentin Hofmann, Lujain Ibrahim, and 23 others. 2025. [Measuring what matters: Construct validity in large language model benchmarks](#). In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.
- Borhane Blili-Hamelin and Leif Hancox-Li. 2023. [Making intelligence: Ethical values in IQ and ML benchmarks](#). In *Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency, FAccT '23*, page 271–284, New York, NY, USA. Association for Computing Machinery.
- Samuel R. Bowman and George Dahl. 2021. [What will it take to fix benchmarking in natural language understanding?](#) In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4843–4855, Online. Association for Computational Linguistics.
- Simin Chen, Yiming Chen, Zexin Li, Yifan Jiang, Zhongwei Wan, Yixin He, Dezhi Ran, Tianle Gu, Haizhou Li, Tao Xie, and Baishakhi Ray. 2025. [Benchmarking large language models under data contamination: A survey from static to dynamic evaluation](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 10080–10098, Suzhou, China. Association for Computational Linguistics.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. [Training verifiers to solve math word problems](#). ArXiv:2110.14168.
- Lee J. Cronbach, Goldine C. Gleser, Harinder Nanda, and Nageswari Rajaratnam. 1972. *The Dependability of Behavioral Measurements: Theory of Generalizability for Scores and Profiles*. Wiley, New York.
- Mostafa Dehghani, Yi Tay, Alexey A. Gritsenko, Zhe Zhao, Neil Houlsby, Fernando Diaz, Donald Metzler, and Oriol Vinyals. 2021. [The Benchmark Lottery](#). ArXiv:2107.07002.
- Ruchira Dhar and Anders Søgaard. 2026. [Evaluation Revisited: A Taxonomy of Evaluation Concerns in Natural Language Processing](#). ArXiv:2604.25923.
- Rotem Dror, Gili Baumer, Marina Bogomolov, and Roi Reichart. 2017. [Replicability Analysis for Natural Language Processing: Testing Significance with Multiple Datasets](#). *Transactions of the Association for Computational Linguistics*, 5:471–486.
- Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. 2018. [The hitchhiker’s guide to testing statistical significance in natural language processing](#). In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1383–1392, Melbourne, Australia. Association for Computational Linguistics.
- Rotem Dror, Segev Shlomov, and Roi Reichart. 2019. [Deep Dominance - How to Properly Compare Deep Neural Models](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2773–2785, Florence, Italy. Association for Computational Linguistics.
- Kenneth Enevoldsen, Isaac Chung, Imene Kerboua, Márton Kardos, Ashwin Mathur, David Stap, Jay Gala, Wissam Siblini, Dominik Krzemiński, Genta Indra Winata, Saba Sturua, Saiteja Utpala, Mathieu Ciancone, Marion Schaeffer, Diganta Misra, Shreeya Dhakal, Jonathan Rystrom, Roman Solomatin, Ömer Veysel Çağatan, and 63 others. 2025. [MMTEB: Massive Multilingual Text Embedding Benchmark](#). In *International Conference on Learning Representations*.
- Stefan Evert. 2004. [Significance tests for the evaluation of ranking methods](#). In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 945–951, Geneva, Switzerland. COLING.

- Manuel Frank and Haithem Afli. 2026a. [The harder text embedding benchmark \(hteb\): Beyond one-dimensional static robustness](#). ArXiv:2605.28190.
- Manuel Frank and Haithem Afli. 2026b. [PTEB: Towards robust text embedding evaluation via stochastic paraphrasing at evaluation time with LLMs](#). In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2832–2851, Rabat, Morocco. Association for Computational Linguistics.
- Timo Freiesleben and Sebastian Zenz. 2025. [The benchmarking epistemology: Construct validity for evaluating machine learning models](#). ArXiv:2510.23191.
- Karan Goel, Nazneen Fatema Rajani, Jesse Vig, Zachary Taschdijan, Mohit Bansal, and Christopher Ré. 2021. [Robustness gym: Unifying the NLP evaluation landscape](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Demonstrations*, pages 42–55, Online. Association for Computational Linguistics.
- Samarth Goel, Reagan Lee, and Kannan Ramchandran. 2025. [SAGE: A Realistic Benchmark for Semantic Understanding](#). In *NeurIPS 2025 Workshop on Evaluating the Evolving LLM Lifecycle: Benchmarks, Emergent Abilities, and Scaling*.
- Kilem Li Gwet. 2008. [Computing inter-rater reliability and its variance in the presence of high agreement](#). *British Journal of Mathematical and Statistical Psychology*, 61(1):29–48.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. [Measuring massive multitask language understanding](#). *Proceedings of the International Conference on Learning Representations (ICLR)*.
- Sture Holm. 1979. [A Simple Sequentially Rejective Multiple Test Procedure](#). *Scandinavian Journal of Statistics*, 6(2):65–70.
- Joseph H. F. James. 2026. [Counting on consensus: Selecting the right inter-annotator agreement metric for NLP annotation and evaluation](#). In *Proceedings of the Fifteenth Language Resources and Evaluation Conference (LREC 2026)*, pages 4434–4446, Palma, Mallorca, Spain. European Language Resources Association (ELRA).
- Ryan Othniel Kearns. 2026. [Quantifying construct validity in large language model evaluations](#). ArXiv:2602.15532.
- Douwe Kiela, Max Bartolo, Yixin Nie, Divyansh Kaushik, Atticus Geiger, Zhengxuan Wu, Bertie Vidgen, Grusha Prasad, Amanpreet Singh, Pratik Ringshia, Zhiyi Ma, Tristan Thrush, Sebastian Riedel, Zeerak Waseem, Pontus Stenetorp, Robin Jia, Mohit Bansal, Christopher Potts, and Adina Williams. 2021. [Dynabench: Rethinking benchmarking in NLP](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4110–4124, Online. Association for Computational Linguistics.
- Ryan Koo, Minhwa Lee, Vipul Raheja, Jong Inn Park, Zae Myung Kim, and Dongyeop Kang. 2024. [Benchmarking cognitive biases in large language models as evaluators](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 517–545, Bangkok, Thailand. Association for Computational Linguistics.
- Yevhen Kostyuk and Kenneth Enevoldsen. 2026. [One prompt is not enough: Instruction sensitivity undermines embedding model evaluation](#). ArXiv:2605.22544.
- Baishi Li, Ta Yu, Kelvin J. L. Koa, and Ke-Wei Huang. 2026. [The Proxy Presumption: From Semantic Embeddings to Valid Social Measures](#). ArXiv:2605.07409.
- Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Cosgrove, Christopher D Manning, Christopher Re, Diana Acosta-Navas, Drew A. Hudson, and 31 others. 2023. [Holistic evaluation of language models](#). *Transactions on Machine Learning Research*. Featured Certification, Expert Certification, Outstanding Certification.
- Zhiyi Ma, Kawin Ethayarajh, Tristan Thrush, Somya Jain, Ledell Wu, Robin Jia, Christopher Potts, Adina Williams, and Douwe Kiela. 2021. [Dynaboard: An evaluation-as-a-service platform for holistic next-generation benchmarking](#). In *Advances in Neural Information Processing Systems*, volume 34, pages 10351–10367. Curran Associates, Inc.
- Iman Mirzadeh, Keivan Alizadeh-Vahid, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. 2025. [GSM-Symbolic: Understanding the Limitations of Mathematical Reasoning in Large Language Models](#). *International Conference on Learning Representations*, 2025.
- Moran Mizrahi, Guy Kaplan, Dan Malkin, Rotem Dror, Dafna Shahaf, and Gabriel Stanovsky. 2024. [State of What Art? A Call for Multi-Prompt LLM Evaluation](#). *Transactions of the Association for Computational Linguistics*, 12:933–949.
- Ishani Mondal and Shweta Bhardwaj. 2026. [Benchmarked Yet Not Measured – Generative AI Should be Evaluated Against Real-World Utility](#). ArXiv:2605.06856.
- Niklas Muennighoff, Nouamane Tazi, Loic Magne, and Nils Reimers. 2023. [MTEB: Massive text embedding benchmark](#). In *Proceedings of the 17th Conference*

- of the European Chapter of the Association for Computational Linguistics, pages 2014–2037, Dubrovnik, Croatia. Association for Computational Linguistics.
- Simon Ott, Adriano Barbosa-Silva, Kathrin Blagec, Jan Brauner, and Matthias Samwald. 2022. [Mapping global dynamics of benchmark creation and saturation in artificial intelligence](#). *Nature Communications*, 13(1):6793.
- Qi Qian, Chengsong Huang, Jingwen Xu, Changze Lv, Muling Wu, Wenhao Liu, Xiaohua Wang, Zhenghua Wang, Zisu Huang, Muzhao Tian, Jianhan Xu, Kun Hu, He-Da Wang, Yao Hu, Xuanjing Huang, and Xiaoqing Zheng. 2026. [Benchmark²: Systematic evaluation of LLM benchmarks](#). ArXiv:2601.03986.
- Inioluwa Deborah Raji, Emily Denton, Emily M. Bender, Alex Hanna, and Amandalynne Paullada. 2021. [AI and the everything in the whole wide world benchmark](#). In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.
- Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. [Squad: 100,000+ questions for machine comprehension of text](#). *Preprint*, arXiv:1606.05250.
- Nils Reimers and Iryna Gurevych. 2018. [Why Comparing Single Performance Scores Does Not Allow to Draw Conclusions About Machine Learning Approaches](#). ArXiv:1803.09578.
- Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. [Beyond accuracy: Behavioral testing of NLP models with CheckList](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online. Association for Computational Linguistics.
- Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. 2024. [Quantifying Language Models’ Sensitivity to Spurious Features in Prompt Design or: How I Learned to Start Worrying About Prompt Formatting](#). In *International Conference on Learning Representations*.
- Shivalika Singh, Yiyang Nan, Alex Wang, Daniel D’souza, Sayash Kapoor, Ahmet Üstün, Sanmi Koyejo, Yuntian Deng, Shayne Longpre, Noah A. Smith, Beyza Ermis, Marzieh Fadaee, and Sara Hooker. 2025. [The leaderboard illusion](#). In *The Thirty-Ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.
- Anders Søgaard, Anders Johannsen, Barbara Plank, Dirk Hovy, and Hector Martínez Alonso. 2014. [What’s in a p-value in NLP?](#) In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 1–10, Ann Arbor, Michigan. Association for Computational Linguistics.
- Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, Agnieszka Kluska, Aitor Lewkowycz, Akshat Agarwal, Alethea Power, Alex Ray, Alex Warstadt, Alexander W. Kocurek, Ali Safaya, Ali Tazarv, and 431 others. 2023. [Beyond the imitation game: Quantifying and extrapolating the capabilities of language models](#). *Transactions on Machine Learning Research*. Featured Certification.
- Anders Søgaard. 2013. [Estimating effect size across datasets](#). In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 607–611, Atlanta, Georgia. Association for Computational Linguistics.
- Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. [GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding](#). In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.
- Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Lingpeng Kong, Qi Liu, Tianyu Liu, and Zhifang Sui. 2024. [Large language models are not fair evaluators](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9440–9450, Bangkok, Thailand. Association for Computational Linguistics.
- Xiao Wang, Qin Liu, Tao Gui, Qi Zhang, Yicheng Zou, Xin Zhou, Jiacheng Ye, Yongxin Zhang, Rui Zheng, Zexiong Pang, Qinzhuo Wu, Zhengyan Li, Chong Zhang, Ruotian Ma, Zichu Fei, Ruijian Cai, Jun Zhao, Xingwu Hu, Zhiheng Yan, and 15 others. 2021. [TextFlint: Unified multilingual robustness evaluation toolkit for natural language processing](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 347–355, Online. Association for Computational Linguistics.
- Chulin Xie, Yangsibo Huang, Chiyuan Zhang, Da Yu, Xinyun Chen, Bill Yuchen Lin, Bo Li, Badih Ghazi, and Ravi Kumar. 2025. [On memorization of large language models in logical reasoning](#). In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pages 2742–2785, Mumbai, India. The Asian Federation of Natural Language Processing and The Association for Computational Linguistics.
- Zihao Yang, Mosh Levy, Yoav Goldberg, and Byron C. Wallace. 2026. [Compared to What? Baselines and Metrics for Counterfactual Prompting](#). ArXiv:2605.01048.

Alexander Yeh. 2000. [More accurate tests for the statistical significance of result differences](#). In *COLING 2000 Volume 2: The 18th International Conference on Computational Linguistics*.

Hugh Zhang, Jeff Da, Dean Lee, Vaughn Robinson, Catherine Wu, Will Song, Tiffany Zhao, Pranav Raja, Charlotte Zhuang, Dylan Slack, Qin Lyu, Sean Hendryx, Russell Kaplan, Michele Lunati, and Summer Yue. 2024. [A Careful Examination of Large Language Model Performance on Grade School Arithmetic](#). *Advances in Neural Information Processing Systems*, 37:46819–46836.

Wenbo Zhang, Hengrui Cai, and Wenyu Chen. 2026. [Beyond the Singular: Revealing the Value of Multiple Generations in Benchmark Evaluation](#). ArXiv:2502.08943.

Hongli Zhou, Hui Huang, Ziqing Zhao, Lvyuan Han, Huicheng Wang, Kehai Chen, Muyun Yang, Wei Bao, Jian Dong, Bing Xu, Conghui Zhu, Hailong Cao, and Tiejun Zhao. 2026. [Lost in Benchmarks? Rethinking Large Language Model Benchmarking with Item Response Theory](#). *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(41):35085–35093.

Kun Zhou, Yutao Zhu, Zhipeng Chen, Wentong Chen, Wayne Xin Zhao, Xu Chen, Yankai Lin, Ji-Rong Wen, and Jiawei Han. 2023. [Don’t make your LLM an evaluation benchmark cheater](#). ArXiv:2311.01964.

Kaijie Zhu, Jindong Wang, Jiaheng Zhou, Zichen Wang, Hao Chen, Yidong Wang, Linyi Yang, Wei Ye, Yue Zhang, Neil Zhenqiang Gong, and Xing Xie. 2024a. [PromptRobust: Towards Evaluating the Robustness of Large Language Models on Adversarial Prompts](#). ArXiv:2306.04528.

Kaijie Zhu, Qinlin Zhao, Hao Chen, Jindong Wang, and Xing Xie. 2024b. [PromptBench: A Unified Library for Evaluation of Large Language Models](#). *Journal of Machine Learning Research*, 25(254):1–22.

A Algorithm

[Algorithm 1](#) gives a task-agnostic specification of the protocol. The generator G returns the base evaluation sets $\{D_r^{(0)}\}_{r=1}^{N_{\text{runs}}}$; depending on the evaluation design, these sets may be resampled across seeds or held fixed while stochastic components vary. All stochastic steps within the run-loop in line 5 to 9 of the algorithm are implicitly based on the corresponding seed. The construct-constrained transformation operator T_C applies each retained perturbation ϕ_j to the base set $D_r^{(0)}$ and returns the perturbed set $D_r^{(j)}$ together with the perturbation-generation fallback rate $\lambda_{r,j}^{\text{gen}}$. Here, $j = 0$ denotes the unperturbed condition and $j = 1, \dots, k$ denote perturbation conditions.

The item-level log $\mathcal{Y}$ is retained so that paired or clustered uncertainty estimates can operate on the appropriate unit of analysis. The aggregation function Agg maps item-level records $\mathcal{Y}_{m,r,j}$ to scalar run-level scores $\mathbf{Z}[m, r, j]$, while $\mathbf{Z}$ stores these aggregate scores for all models, runs, and conditions.

The algorithm distinguishes two failure records. λ^{gen} records perturbation-generation fallbacks, i.e., cases where an attempted perturbation cannot be validly realised and the original input is retained. In contrast, η optionally records model-output failures, such as empty answers or unparseable outputs from the evaluated model; these failures are not counted as perturbation-generation fallbacks. Agg_λ aggregates fallback rates over perturbations within a robustness axis. A_a aggregates perturbation-level model-score vectors into an axis-level model-score vector for robustness axis a . R_{profile} constructs the reported robustness profile, and $R_{\mathcal{H}}$ computes the declared tests with correction Corr applied within the test family $\mathcal{H}$. When at least three models are compared, rank stability is computed with Kendall’s τ . The operator rank_{avg} maps a vector of model scores to ranks using a fixed ranking direction, with tied models assigned the average of the ranks they would occupy. For prompted models, the protocol is run with the primary instruction ι^* ; prompt sensitivity is assessed separately.

B SQuAD Case Study

This section provides short case study of applying MD. It serves as a proof of feasibility and does not provide a full implementation of MD. As such, it does not allow to draw conclusions regarding the performance of the tested models.

B.1 Motivation and Scope

The purpose of this case study is to demonstrate that MD can be applied to a widely common task with perturbations that challenge the established models. Please note that while a real application should include human validation of the perturbations, we only use automatic checks since this is not the full implementation of MD for SQuAD but a case study to demonstrate feasibility.

Construct specification *C*. Construct-preserving perturbations may vary construct-irrelevant aspects of the passage-question pair, such as wording, formatting, surrounding non-

Algorithm 1 Protocol for Measurement-as-Distribution Evaluation

Input: $\mathcal{F} = \{f_1, \dots, f_M\}$, $\mathcal{C}, G, \Phi = \{\phi_1, \dots, \phi_k\}$, $\alpha : \Phi \rightarrow \mathcal{A}, E, N, N_{\text{runs}}$,
 $\mathcal{S} = \{s_1, \dots, s_{N_{\text{runs}}}\}$, *Optional:* $\iota^* \in \mathcal{I} = \{\iota_1, \dots, \iota_q\}$, $\mathcal{H}, \text{Corr}$

```

1:  $(\Phi, V_\Phi) \leftarrow \text{Validate}_{\mathcal{C}}(\Phi)$  # retain valid perturbations, store validation records
2: Define  $T_{\mathcal{C}}, \text{Agg}, \text{Agg}_{\lambda}, R_{\text{profile}}, R_{\mathcal{H}}$ , and  $A_a$  for all  $a \in \mathcal{A}$  # Agg maps item-level records to run-level scores
3: Initialise item-level records  $\mathcal{Y}_{m,r,j}$ , score tensor
    $\mathbf{Z} \in \mathbb{R}^{M \times N_{\text{runs}} \times (k+1)}$ , perturbation-generation fallback rates
    $\lambda^{\text{gen}} \in \mathbb{R}^{N_{\text{runs}} \times k}$ , and optional model-output failure rates
    $\eta \in \mathbb{R}^{M \times N_{\text{runs}} \times (k+1)}$ 

4:  $\{D_r^{(0)}\}_{r=1}^{N_{\text{runs}}} \leftarrow G(\mathcal{C}; N, \mathcal{S})$  # sample base evaluation sets
5: for  $r = 1, \dots, N_{\text{runs}}$  do
6:    $(D_r^{(j)}, \lambda_{r,j}^{\text{gen}}) \leftarrow T_{\mathcal{C}}(\phi_j, D_r^{(0)})$  for all  $j = 1, \dots, k$  # apply perturbations
7:    $(\mathcal{Y}_{m,r,j}, \eta_{m,r,j}) \leftarrow E(f_m, D_r^{(j)}; \iota^*)$  for all  $m = 1, \dots, M, j = 0, \dots, k$  # evaluate models
8:    $\mathbf{Z}[m, r, j] \leftarrow \text{Agg}(\mathcal{Y}_{m,r,j})$  for all  $m = 1, \dots, M, j = 0, \dots, k$  # derive run-level scores
9: end for

10:  $\mathcal{Y} \leftarrow \{\mathcal{Y}_{m,r,j}\}_{m=1, \dots, M; r=1, \dots, N_{\text{runs}}; j=0, \dots, k}$  # collect item-level evaluation records
11:  $\bar{z}_{m,j} \leftarrow \frac{1}{N_{\text{runs}}} \sum_{r=1}^{N_{\text{runs}}} \mathbf{Z}[m, r, j]$  for all  $m = 1, \dots, M, j = 0, \dots, k$  # average scores over runs
12:  $\bar{\lambda}_j^{\text{gen}} \leftarrow \frac{1}{N_{\text{runs}}} \sum_{r=1}^{N_{\text{runs}}} \lambda_{r,j}^{\text{gen}}$  for all  $j = 1, \dots, k$  # average fallback rates
13:  $\bar{\lambda}_a^{\text{gen}} \leftarrow \text{Agg}_{\lambda}(\{\bar{\lambda}_j^{\text{gen}} : \alpha(\phi_j) = a\})$  for all  $a \in \mathcal{A}$  # axis-level fallback rates

14:  $r(f_m) \leftarrow R_{\text{profile}}(\mathcal{Y}, \mathbf{Z}, \lambda^{\text{gen}}, \eta, \alpha; m)$  for all  $m = 1, \dots, M$  # construct robustness profiles
15:  $\mathcal{T}_{\mathcal{H}} \leftarrow R_{\mathcal{H}}(\mathcal{Y}; \mathcal{H}, \text{Corr})$  # declared tests and corrected p-values; returns  $\emptyset$  if  $\mathcal{H} = \emptyset$ 

16: Result  $\leftarrow (\{r(f_m)\}_{m=1}^M, V_\Phi, \lambda^{\text{gen}}, \eta, \{\bar{\lambda}_j^{\text{gen}}\}_{j=1}^k, \{\bar{\lambda}_a^{\text{gen}}\}_{a \in \mathcal{A}}, \mathcal{T}_{\mathcal{H}})$  # store base result tuple without rank-stability

17: if  $M \geq 3$  then # rank stability is reported only for at least three models
18:    $\mathbf{z}^{(j)} \leftarrow (\bar{z}_{1,j}, \dots, \bar{z}_{M,j})$  for all  $j = 0, \dots, k$  # model-score vector for condition  $j$ 
19:    $\tau_j \leftarrow \text{KendallTau}(\text{rank}_{\text{avg}}(\mathbf{z}^{(0)}), \text{rank}_{\text{avg}}(\mathbf{z}^{(j)}))$  for all  $j = 1, \dots, k$  # perturbation-level rank stability
20:    $\mathbf{z}^{(a)} \leftarrow A_a(\{\mathbf{z}^{(j)} : \alpha(\phi_j) = a\})$  for all  $a \in \mathcal{A}$  # axis-level model-score vector
21:    $\tau_a \leftarrow \text{KendallTau}(\text{rank}_{\text{avg}}(\mathbf{z}^{(0)}), \text{rank}_{\text{avg}}(\mathbf{z}^{(a)}))$  for all  $a \in \mathcal{A}$  # axis-level rank stability
22:   Result  $\leftarrow \text{Result} \oplus (\{\tau_j\}_{j=1}^k, \{\tau_a\}_{a \in \mathcal{A}})$  # append rank-stability outputs
23: end if

24: return Result

```

Note: $j = 0$ denotes the unperturbed condition and $j = 1, \dots, k$ denote perturbation conditions. After validation, Φ denotes the retained perturbation procedures, $k = |\Phi|$, and $\mathcal{A}$ contains the retained robustness axes. For models that are not prompted, ι^* is omitted or treated as $\emptyset$. The operator $\oplus$ denotes tuple concatenation.

answer context, or surface noise. They must preserve the information need and the answer relation: the original gold answer must remain valid and recoverable from the provided passage, and the perturbation must not introduce a competing answer or change what the question asks for.

Data generation G . G samples paragraph clusters from the SQuAD v1.1 development set. A paragraph cluster consists of a source paragraph and the question-answer examples sampled from it. Eligible clusters are development-set paragraphs with at least three associated questions. This excludes only a small fraction of paragraphs and ensures that each sampled paragraph contributes the same number of QA items. We sample $N_{\text{para}} = 1,000$ paragraphs, which corresponds to approximately 50% of the paragraph-level SQuAD v1.1 development set.

From each selected paragraph, we sample $Q = 3$ associated questions, yielding

$$N_Q = 1,000 \times 3 = 3,000$$

passage-question-answer items. We choose N_{para} to obtain a sufficiently large sample while keeping compute manageable.

Throughout this example, we use 1337 as a random seed and increment it by one when multiple seeds are required.

Perturbation procedures Φ . We define four deployment-relevant perturbation axes, using one perturbation per axis to keep compute bounded: $k = 4$ and $\Phi = \{\phi_1, \phi_2, \phi_3, \phi_4\}$; see Table 2. Prompt sensitivity is not a primary target of this study and is therefore assessed separately.

Robustness axis	Perturbation	Input	Acceptability criterion
Question formulation	ϕ_1 : paraphrase the question while preserving the information need and expected answer type.	Question	The same information is requested, the same gold answer is required, and no additional constraint is introduced.
Context composition	ϕ_2 : append exactly two non-answer sentences: one topical background sentence and one near-miss distractor mentioning a plausible but incorrect entity, date, quantity, or event.	Passage	No competing correct answer or gold-answer alias is introduced, and the original answer-supporting evidence remains present.
Presentation structure	ϕ_3 : split the passage into sentences and present it in one of two randomly selected formats: bracketed sentence numbers, e.g., “[1]”, or an unnumbered dash-prefixed sentence list.	Passage	After removing the added prefixes and normalising whitespace, passage content and sentence order are recovered exactly; question, gold answer, and the pRobustnessassage-question-answer relation remain unchanged.
Surface noise	ϕ_4 : introduce minor character-level typos into non-protected alphabetic tokens.	Question and passage	The question remains understandable; the evidence remains sufficient; no answer span, named entity, date, quantity, number, negation marker, or token in the answer-supporting sentence is altered.

Table 2: Robustness axes and perturbations for the SQuAD-based QA example.

ϕ_1 and ϕ_2 are generated with the perturbation-generator LLM, g_ϕ = gemma-4-31B-it-AWQ-4bit². We choose this model because it is an open-weight, instruction-tuned model with strong reported static benchmark performance, for example on the Artificial Analysis leaderboard³, while the 4-bit AWQ checkpoint makes local inference feasible on consumer hardware. Perturbation generation is stochastic, using temperature 0.7, top-p 0.95, and maximum output lengths of 256 tokens for question paraphrases and 512 tokens for appended-sentence generation.

ϕ_3 and ϕ_4 are algorithmic. For ϕ_3 , the passage is first split into sentences. For each paragraph cluster and run, we sample one of two presentation formats with equal probability: bracketed sentence numbers, e.g., “[1]”, or an unnumbered dash-prefixed sentence list. The selected format is applied to all QA items drawn from the same paragraph in that run and is logged in the item-level record.

For ϕ_4 , protected tokens are defined before perturbation and include all gold-answer spans, all tokens in answer-supporting sentences, named entities, dates, quantities, numerals, and negation markers. Eligible typo targets are alphabetic tokens of length at least five that are not protected. For each QA item and run, up to $\min(5, \lceil n_{\text{eligible}}/50 \rceil)$ eligible tokens are selected using the seed, and each selected token is perturbed by one character-level op-

eration: adjacent-character swap, single-character deletion, or keyboard-neighbour substitution. If no eligible token exists, the input is left unchanged and the case is logged as a perturbation-generation fallback.

Before model scoring, we validate each perturbation procedure on a subsample. For each ϕ_j , we sample $n_{\text{val}} = 100$ candidate perturbed QA items; for ϕ_3 , candidates are stratified by formatting option. With $k = 4$ perturbations and three annotators per candidate, this yields $4 \times 100 = 400$ candidate perturbations and $400 \times 3 = 1,200$ annotations. Annotators make binary validity judgments according to Table 2, and candidate-level acceptability is decided by majority vote. A perturbation procedure is retained if at least 70 of its 100 candidates are accepted. We report acceptance rates, raw agreement, and Gwet’s AC1 (Gwet, 2008). For the remainder of this section, we assume that all four perturbation procedures pass validation and are retained.

At runtime, each perturbation is passed through an automatic validation method before scoring. We check that outputs are non-empty and parseable, that untargeted fields remain unchanged, and that protected strings or spans are preserved unless explicitly targeted.

Evaluation procedure E. The reference model is fixed before the distribution-based evaluation. However, for this illustrative example, we leave the choice of the models to be compared open. We use $N_{\text{runs}} = 6$ runs. Since perturbation and answer generation are expected to be fast on this dataset,

²<https://huggingface.co/cyankiwi/gemma-4-31B-it-AWQ-4bit>

³<https://artificialanalysis.ai>

evaluation time will still be acceptable with this setting. The unperturbed QA set is identical across runs, while stochastic perturbations are regenerated per run.

Each model is evaluated under five conditions,

$$c \in \{0, \phi_1, \phi_2, \phi_3, \phi_4\},$$

where $c = 0$ is the unperturbed condition. The unperturbed condition uses the same QA inputs across runs, but answer generation is still repeated. Assuming one target model and one predefined reference model, the main evaluation therefore contains

$$2 \times 5 \times 6 \times 3,000 = 180,000$$

scored model outputs. Each generated answer is scored with SQuAD-style token-level F1, taking the maximum F1 over gold answers when multiple gold answers are available.

For each model and evaluation condition, we average SQuAD F1 scores over the selected questions, paragraph clusters, and the $N_{\text{runs}} = 6$ stochastic runs. The total robustness score is the unweighted mean of the four robustness axes, which in this example are identical to the four perturbation scores. Empty outputs or outputs that cannot be normalised into answer text are logged in η and assigned an F1 score of 0 before aggregation.

For each model, we report perturbation deltas relative to its unperturbed score, so that negative values indicate lower performance under the perturbation.

Uncertainty estimation and hypothesis tests.

We estimate uncertainty in the primary analysis with a clustered paired bootstrap over paragraphs using $B = 10,000$ replicates. For each model, condition, paragraph, and question, we first average item-level scores over the $N_{\text{runs}} = 6$ runs. This yields a run-averaged estimate of model performance and reduces seed-specific noise. The runs are not treated as independent observations.

Each bootstrap replicate samples $N_{\text{para}} = 1,000$ paragraph clusters with replacement and keeps all selected questions, evaluation conditions, and paired target-reference model scores together. This preserves paragraph-level dependence and the pairing between models. The resulting two-sided 95% percentile bootstrap intervals quantify paragraph-sample uncertainty for the run-averaged evaluation score. We therefore report run-to-run variation separately as the sample standard deviation across the six runs.

Hypothesis tests use centered two-sided bootstrap p -values. The test family consists of five model comparisons: one comparison between the target model and the reference model on the unperturbed inputs, and one comparison for each of the four perturbation conditions, $\phi_1, \phi_2, \phi_3, \phi_4$. Thus, five p -values are computed and corrected with Holm correction at $\alpha = 0.05$. The total robustness score, within-model perturbation deltas, prompt-sensitivity results, validation outcomes, and fall-back rates are descriptive and are not included in this test family.

Since $M = 2$, rank stability is not computed in this example.

Alternative 1: Using only the passage, provide the brief phrase that answers the question. Do not include explanations.

Alternative 2: Based on the passage, answer the question concisely. Return only the answer text.

The alternative instructions are selectively, manually checked to preserve the requested task and output format. Prompt-sensitivity results are descriptive and we do not report inferential statistics for them. Results for the primary instruction template are re-used in this analysis.

Reporting R . In this example, R reports:

- F1 scores for the target and the predefined reference model under the unperturbed condition and each of the four robustness axes.
- A total robustness score for both models, computed as the equally weighted mean of the four axis-level F1 scores and reported descriptively.
- Paired F1 differences and clustered bootstrap confidence intervals for the unperturbed condition, each perturbation axis, and the total robustness score.
- Five confirmatory p -values, one for the unperturbed condition and one for each perturbation axis, together with Holm-corrected p -values.
- Perturbation acceptance rates, runtime fall-back rates, and inter-rater reliability for each perturbation function ϕ_j using Gwet's AC1.

- Target-model prompt-sensitivity results for three predefined instruction templates⁴, reported descriptively for the unperturbed condition and each perturbation axis.

B.2 Experimental Setup

For all base experiments we used the development set and six random seeds $\in \{1337, \dots, 1442\}$. The prompt sensitivity analysis used three random seeds $\in \{1337, \dots, 1339\}$. All experiments were conducted on 2 x NVIDIA RTX 5090 GPUs, AMD Ryzen 9 9950X CPU, 64GB DDR5-RAM.

B.3 Experimental Results

We assessed how many perturbations of each type passed our programmatic checks. Table 3 shows that question formulation and presentations structure were accepted in all cases. Surface noise was accepted in almost all cases with 99.93% acceptance rate. The outlier was context composition with an acceptance rate of only 82.94% resulting in 3071 original, unperturbed inputs being used. Future work applying this or similar perturbations to SQuAD should examine the reasons and implications. However, this is beyond the scope of our case study.

The robustness profiles of all 14 models are shown in Figure 3. They show mostly stable but not invariant behaviour under perturbation. The average score across perturbations was below the original score for all models. The changes of the average scores were modest in size (between 0.4 and 0.9 points) but not uniform across perturbations and models. Question formulation and context composition often led to the largest reductions compared to the original scores, while changing presentation structure increased the F1 in some cases. Please note that average across perturbations can be viewed as a descriptive summary, but it should not collapse to the results analysis to single dimension by replacing the multi-axis robustness profiles.

Rank correlations confirm that the total model ranking was highly stable with Kendall’s $\tau \in [0.956, 1.000]$ and the lowest Kendall’s τ occurring when the presentation structure was changed (Table 4). Overall, we find that models with clear performance differences remain in the same order while the relative ranking of similarly performing models can change. An example is RoBERTa base

which ranks slightly higher than XLM-R large as measured by the the original scores (91.2 vs. 91.1). But comparing the average perturbation scores the order of the models flips (90.6 vs. 90.7).

Part of the MD protocol is the application of established statistical recommendations for NLP evaluation, including paired uncertainty estimates and correction for multiple comparisons. Following that, we statistically analyse the local ranking changes between RoBERTa base and XLM-R large more closely in Table 5. Based on the original point estimates, RoBERTa base scores 0.12 points higher than XLM-R large. The direction of this difference changes under context composition and presentation structure, for which XLM-R large scores 0.19 and 0.47 points higher, respectively. Under reformulated questions and surface noise, the point estimates again favour RoBERTa base.

However, all paragraph-clustered 95% confidence intervals include zero. None of the five paired model differences is statistically significant, either before or after Holm adjustment assuming the typical significance threshold of 0.05. The displayed ordering of these two models therefore does not provide evidence that either model performs better. Not on the original rankings but also not after perturbation.

This comparison illustrates why small changes in point estimates and model ranks should not be interpreted without their associated uncertainty. Overall, the SQuAD results indicate substantial robustness at the level of global model rankings and modest average changes in F1, but not complete invariance across measurement conditions. The robustness profiles reveal axis-specific responses and local ranking changes that are hidden by evaluation on the original inputs alone, while the paired statistical analysis prevents negligible and uncertain differences from being overinterpreted. In this case study, MD does not overturn the broad conclusion that the strongest models remain strongest. Instead, MD shows here that models with clearly different scores remain in the same order even under perturbations. In contrast, models with very close scores can switch positions and cannot be reliably ranked. For example, RoBERTa base and XLM-R large exchange positions across conditions, but none of their score differences is statistically significant.

⁴The primary instruction template plus the two alternatives.

Perturbation	Attempts	Accepted	Fallbacks	Acceptance rate	Fallback rate
Question formulation	18,000	18,000	0	100.00%	0.00%
Context composition	18,000	14,929	3,071	82.94%	17.06%
Presentation structure	18,000	18,000	0	100.00%	0.00%
Surface noise	18,000	17,988	12	99.93%	0.07%

Table 3: Automatic perturbation acceptance and fallback rates. Accepted variants passed the implemented axis-specific checks; fallbacks use the unchanged original input.



Figure 3: Absolute F1 percentages for the original inputs and each perturbation condition. On the right-hands side the original score and the unweighted average over the perturbations are shown.

Condition	Kendall τ	95% CI
Question formulation	0.978	[0.846, 0.978]
Context composition	0.978	[0.890, 1.000]
Presentation structure	0.956	[0.868, 1.000]
Surface noise	1.000	[0.912, 1.000]
Combined	0.978	[0.912, 1.000]

Table 4: Kendall rank correlation between the original SQuAD model ranking and each perturbed-condition ranking, with paragraph-clustered 95% bootstrap intervals.

Perturbation	RoBERTa base	XLNet-R large	Difference [95% CI]	<i>p</i>	<i>p</i> _{Holm}
Original	91.23	91.12	+0.12 [-0.71, +0.97]	0.7855	1.0000
Question formulation	90.34	89.95	+0.39 [-0.34, +1.17]	0.3066	1.0000
Context composition	90.47	90.67	-0.19 [-1.03, +0.66]	0.6535	1.0000
Presentation structure	91.00	91.47	-0.47 [-1.25, +0.34]	0.2488	1.0000
Surface noise	90.78	90.74	+0.03 [-0.75, +0.83]	0.9402	1.0000

Table 5: Primary SQuAD comparison test. Model columns report absolute F1 scores in %. The difference is RoBERTa base minus XLNet-R large; CIs are paragraph-clustered 95% bootstrap intervals. *p* is the raw p-value; *p*_{Holm} is adjusted for the five comparisons.

Analysis	Method	Observation	# Observations	Goal
Answer scoring	SQuAD-style F1.	One generated answer for one model, question, condition, and seed run.	$2 \times 5 \times 6 \times 3,000 = 180,000$ item-level answer scores in total; $6 \times 3,000 = 18,000$ for each model under each evaluation condition.	Convert generated short answers into item-level scores.
Perturbation validation	Three annotators, binary validity judgments, majority vote, Gwet's AC1, and runtime fallback-rate reporting.	One candidate perturbation for one passage-question-answer item and one perturbation ϕ_j ; for agreement, one annotator rating for one candidate perturbation.	Validation: $4 \times 100 = 400$ candidate perturbations and $400 \times 3 = 1,200$ annotations. Runtime fallback reporting: $4 \times 6 \times 3,000 = 72,000$ perturbed examples.	Check validity, annotation reliability, and whether each ϕ_j produces enough valid variation.
Condition-level scores	Mean aggregation over questions and seed runs for each reported condition.	Item-level F1 scores for sampled SQuAD questions under the unperturbed condition and four perturbation conditions.	For each model under each evaluation condition, $6 \times 3,000 = 18,000$ item-level answer scores are aggregated into one condition-level scalar score. Across two models and five conditions, this yields $2 \times 5 = 10$ condition-level scores.	Estimate model performance under each reported condition.
Total robustness score	Equally weighted mean over robustness-axis F1 scores.	Axis-level F1 scores for one model, after question-level and run-level aggregation. In this example, each axis contains exactly one perturbation, so each axis score equals the corresponding perturbation-condition score.	Four axis-level scores per model, aggregated into one total robustness score; 2 total robustness scores across two models.	Provide a descriptive aggregate robustness summary outside the confirmatory test family.
Model comparison and uncertainty	Paired model differences with a clustered paired bootstrap over SQuAD paragraphs.	One paragraph cluster from the selected main-evaluation sample, keeping associated questions, conditions, and paired target-reference model scores together. Scores are averaged over the six runs before bootstrapping.	The same $N_{\text{para}} = 1,000$ paragraph clusters are used for each of the 5 condition-level paired comparisons and for the descriptive total robustness score comparison; $B = 10,000$ bootstrap replicates are used. Runs are not treated as independent bootstrap observations.	Estimate effect sizes and confidence intervals without treating dependent observations as independent.
Hypothesis tests and correction	Two-sided bootstrap tests with Holm correction.	One predefined target-reference model comparison for one condition, based on run-averaged paragraph clusters.	Each test is based on 1,000 paragraph clusters; Holm correction is applied across the 5 tests.	Test predefined model-comparison hypotheses while controlling the family-wise error rate.
Prompt sensitivity	Auxiliary descriptive analysis over three semantically equivalent instruction templates.	Target-model scores for the same SQuAD questions under different instruction templates, analysed separately from the main target-reference comparison.	$3 \times 5 \times 6 \times 3,000 = 270,000$ target-model item-level answer scores.	Quantify sensitivity to instruction wording separately from the main input-perturbation robustness analysis.

Table 6: Statistical analyses for the SQuAD v1.1 short-answer QA example. Number of observations assumes $M = 2$, $N_{\text{para}} = 1,000$, $Q = 3$ and hence $N_Q = 3,000$, $N_{\text{runs}} = 6$, and five evaluation conditions: one unperturbed condition and four robustness axes with one perturbation procedure each.

C Knights-and-Knaves Case Study

This section describes a brief case study of applying MD to the Knights & Knaves (K&K) (Xie et al., 2025) problem. It serves as a feasibility study with a different configuration of the protocol than the SQuAd case study (Appendix B) and does not provide a full implementation of MD. As such, it does not allow to draw conclusions regarding the performance of the tested models.

Our implementation is based on the code published⁵ by Xie et al. (2025). The authors not only provide a formal representation of the problem instances in their paper but also the code for a stochastic generator of new problem instances, a perturbation function, and an exact solver. These allow changing a single component a problem instance, calculate a solution and verify

its validity, recompute the correct answer, verify the correctness of the generated variant, and evaluate a model on closely related problem instances.

C.1 Task and Construct Specification

In a K&K problem instance, each of N inhabitants is either a *Knight*, who makes a true statement, or a *Knave*, who makes a false statement. Each inhabitant makes exactly one logical statement concerning the roles of one or more inhabitants. The task is to infer which inhabitants are Knights and which are Knaves.

Reusing the notation of Xie et al. (2025), let $B_i \in \{0, 1\}$ denote whether inhabitant i is telling the truth, and let S_i denote the inhabitant's statement. A valid solution satisfies

$$\bigwedge_{i=1}^N (B_i \Leftrightarrow S_i).$$

The released solver exhaustively evaluates the possible Boolean assignments. We retain only problem instances for which exactly one assignment satisfies the complete set of constraints.

The statements are formalised as logical trees if conjunctions, disjunctions, negations, implications, and equivalence operators. Atomic leaves express propositions about whether an inhabitant is telling the truth or lying. The complexity of the problem instances can be controlled by the number of inhabitants N , the maximum statement-tree depth D , and the maximum branching width W . We define

the measured construct $\mathcal{C}$ as a role-assignment with exactly one valid solution.

In the paper the authors provide the following example:

Problem description: A very special island is inhabited only by Knights and Knaves. Knights always tell the truth, and Knaves always lie. You meet 2 inhabitants: Oliver, and Jacob.

Oliver: Oliver is a Knight and Jacob is a Knave.

Jacob: Oliver is a Knight if and only if Jacob is a Knight.

Question: So who is a Knight and who is a Knave?

Formally: Let the boolean propositional variables $B_O \in \{0, 1\}$ and $B_J \in \{0, 1\}$ denote whether Oliver and Jacob are Knights (1) or Knaves (0) respectively. The statements can be denoted as propositional formulas for Oliver (S_O) and Jacob (S_J) as

$$S_O = B_O \wedge \neg B_J,$$

and

$$S_J = B_O \Leftrightarrow B_J.$$

A valid role assignment must satisfy the following:

$$(B_O \Leftrightarrow S_O) \wedge (B_J \Leftrightarrow S_J)$$

because Knights make true and Knaves make false statements. The unique satisfying assignment is

$$B_O = 1 \wedge B_J = 0,$$

hence, Oliver is a Knight and Jacob is a Knave.

A perturbation can, for example, change Oliver's proposition from "Jacob is a Knave" to "Jacob is a Knight" by changing the literal $\neg B_J$:

$$S'_O = B_O \wedge B_J$$

while Jacob's statement remains unchanged:

$$S'_J = S_J = B_O \Leftrightarrow B_J.$$

The perturbed problem instance is therefore

$$(B_O \Leftrightarrow (B_O \wedge B_J)) \wedge (B_J \Leftrightarrow (B_O \Leftrightarrow B_J)).$$

⁵<https://memkklogic.github.io>

This perturbation changes exactly one statement, while the problem instance remains otherwise unchanged. The perturbed problem instance has the unique solution

$$B_O = 1 \wedge B_J = 1.$$

C.2 Mapping to the Measurement-as-Distribution Protocol

The K&K task is mapped to MD as follows:

$\mathcal{C}$: The capability to determine which inhabitants are Knights and which are Knaves in a K&K problem instance with exactly one solution.

$\mathcal{G}$: The K&K generator samples fresh formal problem instances. The verifier rejects problem instances that do not have exactly one solution.

Φ : The perturbation flips exactly one literal in exactly one proposition.

$\mathcal{E}$: The models are evaluated with regard to the parents problems and their perturbations individually considering all instances equal and irrespective of the hierarchical structure of the problem set (i.e., not on a complete problem set of parent and perturbations together but each instance individually since we consider all instances to be equally and individually relevant for the defined construct). Responses are counted as *formatting errors* if the parser could not parse the answer even with basic algorithmic recovery routines. An answer is counted as *role assignment error* if the answer could be parsed but the role assignment is not a correct solution. The models are evaluated with no retries.

$\mathcal{R}$: We report the distribution of error types for parent and across perturbed problem instances. Moreover, verification results are reported for $\mathcal{G}$ and Φ .

C.3 Base problem instance Generation

We generate a fresh set of 100 base problem instances using the released K&K sampler. All 100 problem instances have $N = 16$ inhabitants. The maximum statement-tree depth (D) and branching width (W) are fixed at $D = 3$ and $W = 2$.

A generated candidate is admissible if and only if the solver returns exactly one role assignment and the candidate is not a formal duplicate of an already retained problem instance.

The final clean set is selected jointly with the perturbation panel described below. Specifically, a

Parameter	Value
Inhabitants per instance	16
Maximum depth (D) of instances	3
Maximum width (W) of instances	2
Number of generated parent instances	100
Number perturbations per parent	5
Evaluated models	2

Table 7: K&K Case study specification.

base problem instance is retained only when five valid and formally distinct leaf variants can be generated for it.

C.4 Perturbations

For each clean parent problem instance, the perturbation procedure randomly selects one speaker’s logical statement, selects one literal within that statement, and replaces it with another valid literal. A candidate variant is accepted only when all of the following conditions hold:

1. The perturbed problem instance differs from the parent problem instance by replacing exactly one literal in exactly one speaker’s logical statement.
2. The names and ordering of the inhabitants, the role labels, and the number and ordering of the logical statements remain unchanged. All logical operations, groupings, and literals other than the replaced literal also remain unchanged.
3. The exact solver returns exactly one satisfying role assignment for the perturbed problem instance.
4. This reference role assignment differs from that of the corresponding base problem instance for at least one inhabitant.
5. The formal statements of the perturbed problem instance are not identical to those of any other retained base or perturbed problem instance.

Every perturbed set contains one variant of each of the same 100 parent problem instances. The resulting evaluation panel therefore contains 600 items: 100 clean problem instances and 500 leaf-perturbed problem instances.

C.5 Experimental Setup

We evaluate the open-weight generative LLMs Gemma-4-31B and Qwen-3.8-27B. The problem specification is detailed in Table 7. The evaluation is zero-shot and uses the following hyperparameter settings.

Gemma-4-31B⁶:

- Checkpoint: google/gemma-4-31B-it-qat-w4a16-ct
- Temperature: 1.0
- Top- p : 0.95
- Top- k : 64
- Min- p : 0.0
- Presence penalty: 0.0
- Frequency penalty: 0.0
- Repetition penalty: 1.0
- Maximum output length: 7,680 tokens
- Maximum context length: 8,192 tokens
- Generation seed: 20260803
- Thinking enabled
- Batch size: 8
- Maximum active sequences: 4
- Tensor parallelism: 1
- Pipeline parallelism: 1
- GPU-memory utilization: 0.92

Qwen-3.8-27B⁷:

- Checkpoint: Qwen/Qwen3.8-27B-FP8
- Temperature: 1.0
- Top- p : 0.95
- Top- k : 20
- Min- p : 0.0

⁶<https://huggingface.co/google/gemma-4-31B-it-qat-w4a16-ct>

⁷<https://huggingface.co/Qwen/Qwen3.8-27B-FP8>

- Presence penalty: 0.0
- Frequency penalty: 0.0
- Repetition penalty: 1.0
- Maximum output length: 32,768 tokens
- Maximum context length: 65,536 tokens
- Generation seed: 20260803
- Thinking enabled
- Reasoning effort: xhigh
- Batch size: 2
- Maximum active sequences: 2
- Tensor parallelism: 2
- Pipeline parallelism: 1
- GPU-memory utilization: 0.90

We use the following prompt:

K&K Default Prompt

You solve Knights-and-Knaves logic problem instances. Every inhabitant is exactly one of two roles: a Knight, who always makes a true statement, or a Knave, who always makes a false statement. Determine the role of every named inhabitant. Solve the problem instance carefully. Your final line must begin with FINAL_JSON: and contain one JSON object mapping every inhabitant's name to exactly "Knight" or "Knave". Include every inhabitant exactly once and include no other keys in that JSON object.

The natural-language problem instance is supplied as the user message. The system instruction is held identical across models, items, and conditions. Only the text of the problem instance changes.

Each model receives all 600 evaluation items, producing 600 responses per model and 1,200 responses in total. lean and perturbed items are combined in one deterministically shuffled input order so that conditions are not processed in separate blocks. The same item order is used for both models.

All experiments were conducted on the following hardware: 2 x NVIDIA RTX 5090 GPUs, AMD Ryzen 9 9950X CPU, 64GB DDR5-RAM.

Type	Accepted / proposed	Acceptance rate
Parents	100/355	28.2%
Variants	500/3,997	12.5%

Table 8: K&K observed proposal acceptance rates. Parent problem-instance acceptance was assessed without considering variant availability. Variant acceptance was conditional on an accepted parent problem instance.

Instances	Role		Total
	assignment	Formatting	
<i>Gemma-4-31B</i>			
Parents	3	0	3 (3.0%)
Perturbations	12	0	12 (2.4%)
Total	15	0	15 (2.5%)
<i>Qwen-3.8-27B</i>			
Parents	1	1	2 (2.0%)
Perturbations	5	4	9 (1.8%)
Total	6	5	11 (1.8%)

Table 9: Role assignment and unrecoverable formatting errors on parent and perturbed problem instances in the K&K case study.

C.6 Experimental Results and Discussion

Overall, the generated generated 355 parent problem of which 28.2% were accepted. To obtain the corresponding 500 perturbations, 3997 perturbed examples were generated (see Table 8).

Overall, Gemma-4-31B solved 585 out of 600 problem instances correctly and Qwen-3.8-27B solved 589 (Table 9). However, as the table shows the error profiles of the two models are distinctly different: all of Gemma’s 15 errors are role assignment errors while five out of Qwen’s nine errors are formatting errors (i.e., the model failed to provide an answer that could be parsed).

As an additional analysis we evaluated Gemma-4-27B on $n = 20$ parent problem instances (each with 5 perturbations) using the following paraphrased prompt:

K&K Prompt Variant

Work out the solution to this Knights-and-Knaves logic puzzle. Each inhabitant has exactly one role: knights always tell the truth, while knaves always make false statements. Reason carefully and identify the role of every named inhabitant. Conclude with a final line beginning FINAL_JSON:, followed by a single JSON object. Use every inhabitant’s name exactly once as a key, map each key to

either "knight" or "knave", and include no additional keys.

As intended, the K&K case study demonstrates the feasibility of verifier backed task with perturbations that are not label-preserving. The case studies was not designed draw conclusions regarding model performance and does, therefore, not report CIs or p-values. For the same reason we refrain from an statistical analysis overview which has been exwmplified by the SQuAD case study already. We do not make general claims about how the evaluated models perform on the K&K task. Moreover, future applications that are intended to measure model performance and robustness should include more challenging problem instances or perturbations.