ConsistencyChecker: Tree-based Evaluation of LLM Generalization Capabilities

Anonymous ACL submission

Abstract

Evaluating Large Language Models (LLMs) requires effective methods to assess how well they maintain semantic consistency across multiple transformations. Traditional methods like self-consistency often fail to capture subtle semantic errors that emerge during multi-step tasks. To address this, we introduce *ConsistencyChecker*, a benchmark-free evaluation framework that evaluates LLMs' ability to preserve semantic consistency during multi-step processes. Our approach is based on the concept of a self-consistency tree, where each node represents a state of the text after a transformation (*e.g.*, translation, code modification, paraphrasing), and each edge represents the transformation itself. By constructing self-consistency trees, we measure how accurately the model maintains the original meaning across these changes. ConsistencyChecker quantifies an LLM's reliability in retaining critical information by analyzing semantic preservation between nodes at different tree depths. This approach provides insights into model generalization capabilities without requiring extensive resources. Experiments show that ConsistencyChecker can accurately measure the generalization ability of models with various sizes on translation and coding tasks without the need for building benchmarks. By identifying scenarios where models maintain or lose semantic fidelity, ConsistencyChecker offers a practical tool for understanding LLM robustness in real-world applications.

1 Introduction

Large Language Models (LLMs) have emerged as transformative tools in artificial intelligence, demonstrating remarkable capabilities across diverse tasks, including natural language understanding (Minaee et al., 2024), generation (Minaee et al., 2024), and complex reasoning (Brown et al., 2020; Chowdhery et al., 2024). Trained in a vast corpus of text data, these models have achieved unprecedented performance in areas ranging from machine translation (Qian et al., 2024; Zhu et al., 2024) to code generation (Jiang et al., 2024; Wang et al., 2023), and from creative writing (Wang et al., 2024a; Marco et al., 2024) to scientific analysis (Zhang et al., 2024). Recent architectural advances, coupled with increases in model scale and training data, have led to LLMs that can engage in sophisticated dialogue, solve complex problems, and exhibit emergent abilities (Wei et al., 2022).

However, as LLMs are increasingly deployed in real-world applications, evaluating their generalization capabilities—particularly in low-resource settings - has become a critical challenge. Dataset-based evaluation methods (*e.g.*, using HumanEval to measure code generation ability) often rely on extensive benchmark datasets and human annotations, which are costly, time-intensive, and prone to limitations such as benchmark contamination and dataset bias (Geirhos et al., 2020; Bowman and Dahl, 2021; Magar and Schwartz, 2022). Moreover, these dataset-centric approaches struggle to assess how well models maintain semantic consistency across diverse transformations, a key indicator of their robustness and reliability in practical scenarios (Callison-Burch et al., 2006; Zhang* et al., 2020). This gap is particularly pronounced in low-resource settings, where access to large-scale benchmarks and human evaluators is limited.

Recent work has explored alternative evaluation approaches, including human-in-the-loop frameworks (Chiang et al., 2024; Gao et al., 2023) and LLM-as-a-judge paradigms (Zheng et al., 2023), which combine human flexibility with automated scalability. However, these methods still often depend on task-specific examples or external references, limiting their applicability in low-resource or domain-agnostic settings. Additionally, the challenge of hallucination—where models generate inconsistent or factually incorrect re-

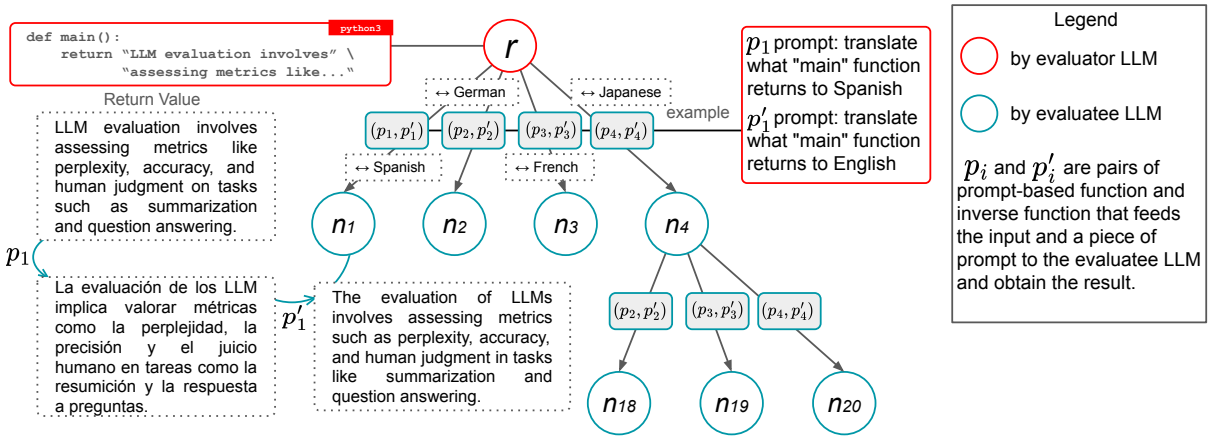

Figure 1: **Overview of the ConsistencyChecker Framework.** This illustrates the evaluation process in ConsistencyChecker, where a tree-based self-consistency approach is used to evaluate LLMs. The root node (r) is a piece of text generated by the evaluator model given constraints, and subsequent levels. $(v_1, v_2, \dots)$ represent nodes corresponding to nodes of text going through the transform-reverse functions executed by the evaluatee model(*e.g.*, round-trip translation (van Zaanen and Zwarts, 2006)), where the functions are generated by the evaluator.

sponses—remains a significant barrier to reliable evaluation (Waldo and Boussard, 2024). While dataset-free metrics like self-consistency scores have emerged, they primarily focus on single-step evaluation and fail to capture nuanced semantic discrepancies in multi-step tasks. Moreover, current evaluation metrics such as self-consistency scores tend to focus on trivial robustness checks rather than identifying the subtle semantic variations that emerge during complex multi-step reasoning.

Inspired by these advances, we introduce ConsistencyChecker, a novel benchmark-free framework for evaluating LLMs' generalization capabilities. ConsistencyChecker operates without requiring predefined benchmarks or reference datasets, enabling rapid assessment of model performance across arbitrary domains. At its core, ConsistencyChecker proposes the concept of self-consistency trees where each node represents a transformation step (*e.g.*, translation, paraphrasing), each edge denotes a semantic-preserving operation, and each path from the root node to any leaf node represents a multi-step transformation sequence. By constructing these self-consistency trees, our work measures how accurately the model preserves the original meaning across multiple transformations.

By analyzing semantic preservation between nodes at different tree depths, ConsistencyChecker quantifies an LLM's reliability in retaining critical information through iterative transformations. This approach enables comprehensive evaluation of LLMs' generalization ability in a multi-step manner without relying on costly benchmark construction or annotated datasets. The framework further identifies subtle semantic discrepancies that emerge during complex reasoning processes, providing a robust and resource-efficient alternative to traditional dataset-dependent evaluation paradigms.

Our approach addresses several fundamental challenges in LLM evaluation. First, as a benchmark-free method, it eliminates the need for domain-specific benchmarks or annotated datasets, enabling immediate application to new tasks or domains. This ensures evaluation adaptability across evolving models while avoiding traditional pitfalls like dataset overfitting and benchmark contamination. Second, through multi-step semantic evaluation across transformation chains, ConsistencyChecker analyzes robustness by measuring how models preserve critical information through iterative changes—providing deeper insights into generalization capabilities.

Our experiments demonstrate that ConsistencyChecker effectively ranks LLM performance. In code equivalence preservation, Qwen 2.5 32B achieves the highest consistency score with an L3 of 85.1. For cross-lingual translation consistency, GPT 4o Mini leads with an L3 of 98.0. Surprisingly, we find that model size does not strictly correlate with robustness - Qwen 2.5 32B outperforms Qwen 2.5 72B in AI-assisted programming tasks despite having fewer parameters.

2 Related Works

Software Verification and Formal Methods. The development of reliable systems has long relied on formal verification techniques such as model checking (Clarke et al., 1999), theorem proving (Nipkow et al., 2002), and satisfiability modulo theories (SMT) solvers (Barrett and Tinelli,

2018). While these methods provide rigorous guarantees, they typically require significant computational resources and domain-specific specifications. Our work draws inspiration from the iterative refinement principles of model checking but adapts them to evaluate LLMs through lightweight, self-referential transformations. This shift enables scalable verification without predefined specifications, aligning with the challenges of evaluating generative models in resource-constrained scenarios.

The rise of LLMs for code generation has spurred benchmarks like CodeXGLUE (Lu et al., 2021) and ProblemSolving (Hendrycks et al., 2021), which evaluate capabilities such as code completion and problem-solving. However, these benchmarks require curated datasets of human-written code solutions, limiting their applicability to novel or under-resourced programming tasks. Recent work highlights the importance of code robustness—ensuring models generate functionally equivalent outputs across stylistic variations (Chen et al., 2021). Our work operationalizes this idea through bidirectional code transformations (*e.g.*, refactoring, language translation), measuring whether semantic similarity is preserved.

Translation Evaluation and Round-Trip Consistency. Machine translation evaluation has evolved from early rule-based systems (Weaver, 1952) to modern transformer-based approaches (Vaswani et al., 2017). Traditional metrics like BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), and TER (Snover et al., 2006) rely on reference translations, while benchmarks like WMT (Barrault et al., 2019) depend on human-annotated parallel corpora—resources often unavailable in low-resource languages or specialized domains. Round-trip translation, where text is translated between languages and back (van Zaanen and Zwarts, 2006), provides a benchmark-free consistency check analogous to our transformation chains. ConsistencyChecker generalizes this concept, extending it beyond translation to arbitrary semantic-preserving operations (*e.g.*, code refactoring, summarization), thereby enabling evaluation in scenarios where parallel data or reference outputs are absent.

Robustness and Consistency Evaluation Recent advancements in LLM evaluation emphasize structured frameworks to address consistency across multi-step interactions. While verbalized confidence scoring methods assess self-awareness through explicit prompts (Yang et al., 2024), they often conflate calibration with multi-step stability. The Divide-Conquer-Reasoning approach introduces hierarchical evaluation by decomposing comparisons into granular analyses, enabling precise detection of inconsistencies (Cui et al., 2024). This aligns with dynamic frameworks like MT-Eval that benchmark recollection and refinement capabilities in multi-turn workflows (Kwan et al., 2024). Such methods contrast with traditional evaluations that miss semantic drift in iterative transformations, particularly evident when applying adversarial prompts to preserve functional similarity (Wang et al., 2024b).

3 Preliminary

A common use case for large language models (LLMs) involves a user providing an initial input (*e.g.*, a code snippet) along with a prompt requesting specific alterations, which the LLM then executes. In real-world scenarios, users often impose multiple requirements through multi-step interactions, introducing complexity and nuance at each stage. To systematically evaluate LLM consistency through such iterative interactions, we propose a tree-structured framework that supports hierarchical transformations and multi-step analysis. This structure enables comprehensive evaluation through three key properties: (1) hierarchical organization of code variants, (2) controlled multi-step transformations, and (3) systematic consistency measurement through path analysis.

Self-consistency tree A self-consistency tree $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ is a directed tree structure that simulates multi-step user-LLM interactions. Each node $v \in \mathcal{V}$ represents a distinct code implementation, while edges $e \in \mathcal{E}$ represent transformations applied by LLMs through prompt operations. The tree structure enables systematic evaluation of output consistency across multiple transformation steps while preserving functional similarity through shared test inputs.

Node Each node $v = (d, c, \mathcal{I}, l)$ represents an executable code implementation through four components: $d \in \sum^*$ denotes the natural language task description, $c \in \sum^*$ represents the executable code implementation, $\mathcal{I} = [i_1, ..., i_n]$ specifies an ordered list of test inputs where each $i_k \in \sum^*$, and $l \in \sum^*$ indicates the tag of a programming language (*e.g.*, "python3"). Here $\sum^*$ denotes all

potential combinations of strings. Nodes maintain functional similarity verification through shared test inputs $\mathcal{I}$ across transformations.

Operation An operation is a prompt-driven code transformation function executed by an LLM. We design operation pairs p and p' that ideally nullify each other's effects (*e.g.*, "add logging" followed by "remove logging"). Formally, an operator is:

$$f_p : c \to c' \tag{1}$$

where c is input code, p a transformation prompt, and c' the transformed code. Operation pairs test the LLM's transformation consistency.

Edge Edges enforce functional similarity through invariant test inputs while tracking code evolution via successive LLM operations. Formally, an edge $e_{ij} = (v_i, v_j)$ connects nodes $v_i = (d_i, c_i, \mathcal{I}_i, l_i)$ and $v_j = (d_j, c_j, \mathcal{I}_j, l_j)$ if and only if:

$$\begin{cases} d_i = d_j, \;\; \mathcal{I}_i = \mathcal{I}_j, \;\; l_i = l_j, \\ c_j = f_{p'}(f_p(c_i)). \end{cases} \tag{2}$$

Path A path in the self-consistency tree can be formally written as $P = (v_1, ..., v_k)$. It connects a node to any descendant through edges. The length of one path equals the number of transformations applied (depth difference between nodes).

Forest A forest $\mathcal{F} = \{\mathcal{T}_1, ..., \mathcal{T}_m\}$ combines multiple trees generated from different initial nodes. This enables comparative analysis of LLM consistency across varied tasks, languages, and transformation sequences.

4 ConsistencyChecker: Tree-based LLM evaluation for Generalization Ability

Traditional single-tree evaluation faces inherent limitations through three primary biases: task-specific difficulty variance (*e.g.*, code translation being simpler than algorithm reconstruction), oversensitivity to prompt phrasing variations, and implementation singularities from unique root node configurations. Our forest-based approach addresses these challenges through strategic diversification - roots sample diverse task types across the problem space, transformation prompts vary per tree to reduce phrasing bias, and architectural configurations differ across trees to avoid implementation lock-in. This multi-axis variation produces robust consistency estimates that better reflect real-world LLM deployment scenarios.

Similarity score for node-pair For a pair of nodes v_i, v_j, where $v_i = (d, c_i, \mathcal{I}, l)$ and $v_j = (d, c_j, \mathcal{I}, l)$, that shares the same task description d, test inputs $\mathcal{I}$, and language l, semantic similarity is computed through a three-stage process. First, execute both codes to get outputs $\mathcal{O}_i = \text{exec}(c_i, \mathcal{I})$ and $\mathcal{O}_j = \text{exec}(c_j, \mathcal{I})$. Then, convert outputs to standardized string representations $s_i = \text{str}(\mathcal{O}_i)$ and $s_j = \text{str}(\mathcal{O}_j)$. Finally, functional similarity can be measured as the cosine similarity based on embedding. The whole process can be written as:

$$\begin{aligned} s(v) &= \text{Str}(\mathcal{O}(v)) \\ \mathcal{E}(v) &= \text{Emb}(s(v)) \\ \text{sim}(v_i, v_j) &\coloneqq \frac{\mathcal{E}(v_i) \cdot \mathcal{E}(v_j)}{\|\mathcal{E}(v_i)\| \|\mathcal{E}(v_j)\|} \end{aligned} \tag{3}$$

where $\text{Emb}(\cdot)$ generates semantic embeddings through a pretrained language model. This approach recognizes equivalent functionalities with divergent syntactic outputs (*e.g.*, `"42"` vs `"42.0"`) while providing granular similarity scores in $[0, 1]$.

Consistency score for path To evaluate whether iterative transformations preserve core functionality, we measure the end-to-end similarity between the initial and final nodes in a path. For a path $P = (v_1, \cdots, v_k)$, the consistency score is:

$$C(P) \coloneqq \text{sim}(v_1, v_k) \tag{4}$$

This metric measures whether the final output v_k remains functionally equivalent to the original v_1 after multiple modifications. For example, applying "add logging functionality" after "remove logging functionality" should preserve the sorting functionality, which this score captures.

Consistency score for tree To evaluate an LLM's robustness across all possible transformation sequences of fixed length, we compute path consistency for every N-step path in the tree. For depth parameter N, this metric aggregates all paths $\mathcal{P}_N = \{(v_i, \cdots, v_j) \mid \text{length}(P) = N\}$:

$$C(\mathcal{T}) \coloneqq \frac{1}{|\mathcal{P}_N(\mathcal{T})|} \sum_{P \in \mathcal{P}_N(\mathcal{T})} C(P) \tag{5}$$

This measures average functional preservation across all N-transformation sequences, capturing both cumulative error propagation and path-specific divergence patterns. For example, a tree with a high L-3 score but a low L-5 score indicates consistent performance for short transformation chains but degradation in longer sequences.

Algorithm 1 Tree Generation Algorithm

Require: Root node r, max depth D, operation pairs $((p_1, p_1'), \cdots, (p_k, p_k'))$
Ensure: Tree $\mathcal{T} = (\mathcal{V}, \mathcal{E})$

$\mathcal{V} \leftarrow \{ r \}$
$\mathcal{E} \leftarrow \emptyset$
for $d \leftarrow 1$ to D **do**
  $U \leftarrow \{ v \in \mathcal{V} \mid \mathcal{D}(v) = d - 1 \}$
  for each $v \in U$ **do**
    for $i \leftarrow 1$ to k **do**
      $c' \leftarrow f_{p_i}(v.c)$
      $c'' \leftarrow f_{p_i'}(c')$
      Let $v' \leftarrow (d, c'', v.\mathcal{I}, v.l)$
      $\mathcal{V} \leftarrow \mathcal{V} \cup \{ v' \}$
      $\mathcal{E} \leftarrow \mathcal{E} \cup \{ (v, v') \}$
    end for
  end for
end for
return $(\mathcal{V}, \mathcal{E})$

Consistency score for forest To assess cross-task robustness, we compute two key statistics from the forest $\mathcal{F} = \{\mathcal{T}_1, ..., \mathcal{T}_k\}$:

$$C_\mu(\mathcal{F}) := \frac{1}{|\mathcal{F}|} \sum_{\mathcal{T} \in \mathcal{F}} C(\mathcal{T}),$$
$$C_\sigma(\mathcal{F}) := \sqrt{\frac{1}{|\mathcal{F}|} \sum_{\mathcal{T} \in \mathcal{F}} \Big(C(\mathcal{T}) - \mu_{\mathcal{F}} \Big)^2}. \quad (6)$$

where $\mu_{\mathcal{F}}$ represents average consistency within the task, and $\sigma_{\mathcal{F}}$ measures evaluation stability, where a larger $\sigma_{\mathcal{F}}$ means the average consistency score is more reliable. This separation reveals whether an LLM achieves high consistency uniformly ($\sigma_{\mathcal{F}} \approx 0$) or excels only in specific circumstances ($\sigma_{\mathcal{F}} \gg 0$).

ConsistencyCheck In ConsistencyCheck, the evaluator model first generates the root node r and a list of operation pairs $(p_1, \cdots, p_k)$ conditioned on the root node, based on meta-prompts that describes the desired properties in the root node and operations. We then use the evaluatee model to build a self-consistency tree given root node r and k pairs of operations, $(p_1, p_2, \cdots, p_k)$. The tree generation process is described in Algorithm 1.

5 Experimental Setting

5.1 Models

We use a total of 4 popular open-source LLMs for evaluation: Qwen 2.5 7B, Qwen 2.5 72B, LLaMA 3.1 8B, and LLaMA 3.1 70B. All 4 models are used as evaluators, which generate the root nodes of self-consistency trees. These models are also being evaluated (as evaluatee) using these generated root nodes. For better prompt-following, all of the models being used are instruction-tuning versions.

5.2 Evaluation Metrics

For each task, we computed these metrics across multiple levels of transformation (denoted as L1, L2, and L3). The results are reported as mean scores with standard deviations to account for variability across different evaluation instances.

Embedding-based metric To comprehensively evaluate the models, we employed the following metrics. First, we used the embedding similarity, computed using the NV-Embed-v2 [1], to measure cosine similarity between the embeddings of generated outputs. This metric captures semantic similarity, even when the syntactic structure differs.

N-gram-based metric We utilized the BLEU score, which measures the precision of n-gram overlaps between generated and reference outputs, providing a quantitative measure of text similarity.

5.3 Tasks and Meta Prompts

Our evaluation framework is built around two tasks, each defined by a meta prompt that generates a set of root nodes and paired operations. These operations expand into trees during benchmarking, enabling a structured evaluation of model performance across varying levels of complexity.

For the **translation task**, the root node is a randomly sampled English paragraph of approximately 400 words in length. The operations involve translating this paragraph into and from another language. We sample three languages from an evaluator function, which are typically French, Spanish, and German. This setup ensures that the task captures the nuances of multilingual translation across diverse linguistic contexts.

For the **AI-assisted programming task**, the root node consists of a LeetCode-Hard style programming problem, its solution, and 20 accompanying test cases. The operations are based on the root code and involve transformations that request the code to be implemented in a different but valid and equivalent way. This design allows us to evaluate the model's ability to generate functionally correct

[1] https://huggingface.co/nvidia/NV-Embed-v2

Table 1: **Cross-lingual Machine Translation Consistency Evaluation.** Scores(%) evaluate LLMs' ability to attain semantics after translating across different languages. The evaluated models vary in model families, scale (0.5B-72B) and path length (1-3).

Evaluatee	L1	L2	L3
GPT 4o mini	**99.2** $\pm$ 0.0	**98.7** $\pm$ 0.0	**98.0** $\pm$ 0.0
Qwen 2.5 1.5b	91.9 $\pm$ 0.1	86.5 $\pm$ 0.1	80.3 $\pm$ 0.5
Qwen 2.5 7b	96.4 $\pm$ 0.1	93.5 $\pm$ 0.3	90.0 $\pm$ 0.8
Qwen 2.5 14b	97.4 $\pm$ 0.0	96.1 $\pm$ 0.1	94.7 $\pm$ 0.1
Qwen 2.5 32b	97.6 $\pm$ 0.1	97.4 $\pm$ 0.0	96.4 $\pm$ 0.0
Qwen 2.5 72b	99.1 $\pm$ 0.0	98.3 $\pm$ 0.0	97.2 $\pm$ 0.0
LLaMA 3.1 8b	89.5 $\pm$ 0.1	75.7 $\pm$ 1.6	67.5 $\pm$ 3.0
LLaMA 3.1 70b	87.4 $\pm$ 0.7	80.7 $\pm$ 1.1	71.9 $\pm$ 3.2

Table 2: **Code Equivalence Preservation in AI-Assisted Programming Tasks Evaluation.** Scores(%) evaluate LLMs' ability to attain code equivalence and functionality when performing algorithm transformations multiple times.

Evaluatee	L1	L2	L3
GPT 4o mini	90.6 $\pm$ 0.2	84.7 $\pm$ 0.6	76.5 $\pm$ 2.7
Qwen 2.5 1.5b	90.2 $\pm$ 0.2	80.0 $\pm$ 0.6	63.4 $\pm$ 1.4
Qwen 2.5 7b	85.4 $\pm$ 0.2	80.2 $\pm$ 0.2	71.7 $\pm$ 0.4
Qwen 2.5 14b	90.3 $\pm$ 0.2	83.1 $\pm$ 0.6	79.9 $\pm$ 1.0
Qwen 2.5 32b	**91.1** $\pm$ 0.4	**88.3** $\pm$ 0.7	**85.1** $\pm$ 1.1
Qwen 2.5 72b	90.8 $\pm$ 0.2	85.3 $\pm$ 0.5	77.0 $\pm$ 1.9
LLaMA 3.1 8b	88.1 $\pm$ 0.2	76.0 $\pm$ 0.5	60.4 $\pm$ 1.0
LLaMA 3.1 70b	87.9 $\pm$ 0.4	82.7 $\pm$ 0.6	83.5 $\pm$ 1.0

and semantically consistent code under varying levels of abstraction and complexity.

Both tasks are structured as trees, where the root node serves as the starting point, and the operations represent branches that expand into increasingly complex transformations. This hierarchical approach provides a systematic way to measure model performance across multiple levels of difficulty and variation.

In each task, each evaluator model generated 10 root nodes, which are then used consistently to evaluate all LLMs.

6 Experimental Results

The results of our experiments are summarized in Tables 1 and Table 2. Table1 presents the evaluation metrics for the translation task, while Table 2 focuses on the AI-assisted programming task. Each table reports the mean scores and standard deviations for embedding similarity.

In the **translation task**, the results indicate that larger models consistently outperform their smaller counterparts, particularly in handling longer sequences of linguistic transformations (L3). For instance, the Qwen 2.5 72B model achieves a consistency score of 99.1 at L1, 98.3 at L2, and 97.2 at L3, which are significantly higher than the scores of the Qwen 2.5 1.5B model (91.9 at L1, 86.5 at L2, and 80.3 at L3). Similarly, the LLaMA 3.1 70B model shows an improvement over the LLaMA 3.1 8B model, with scores increasing from 89.5 to 87.4 at L1, from 75.7 to 80.7 at L2, and from 67.5 to 71.9 at L3.

In the **AI-assisted programming** task, larger models also demonstrate superior performance in generating functionally correct and semantically consistent code across all levels of transformation. The Qwen 2.5 32B model achieves the highest scores at all levels: 91.1 at L1, 88.3 at L2, and 85.1 at L3, compared to the Qwen 2.5 1.5B model's scores of 90.2 at L1, 80.0 at L2, and 63.4 at L3. The LLaMA 3.1 70B model also shows a notable improvement over the LLaMA 3.1 8B model, with scores increasing from 88.1 to 87.9 at L1, from 76.0 to 82.7 at L2, and from 60.4 to 83.5 at L3.

These findings underscore the importance of model scale and architecture in achieving robust and consistent performance across diverse tasks and transformation levels. The detailed results are discussed in the following sections.

7 Ablation Study

Ablation under path length. Figure 2 shows that longer paths have lower consistency scores under increasing levels of transformation complexity. For instance, in the cross-lingual translation task, the LLaMA 3.1 8B model exhibits a decrease in consistency scores from 89.5 at L1 to 75.7 at L2 and further down to 67.5 at L3. Similarly, the Qwen 2.5 72B model shows a decline from 99.1 at L1 to 98.3 at L2 and 97.2 at L3. This trend is consistent across most of the models of varying families and sizes evaluated. In the AI-assisted programming tasks, the Qwen 2.5 7B model also demonstrates a reduction in scores from 85.4 at L1 to 80.2 at L2 and 71.7 at L3, further illustrating the impact of path length on consistency.

Ablation under evaluator model. As shown in Table 3 and Table 4, different evaluator models (Qwen 2.5 7B, Qwen 2.5 72B) provide similar rankings for different models. For instance, in the translation task, the Qwen 2.5 7B evaluator model gives scores of 90.1 at L1, 82.8 at L2, and 76.8 at L3,

Table 3: **Cross-lingual Translation Consistency Under Iterative Transformations.** Consistency scores(%) across model scales (7B-72B) and transform complexity (L1-L3) using embedding similarity and text overlap metrics

Evaluator / Evaluatee	Qwen 2.5 7B			Qwen 2.5 72B		
	L1	L2	L3	L1	L2	L3
Qwen 2.5 7b	**90.1** $\pm$ 0.9	82.8 $\pm$ 3.0	76.8 $\pm$ 5.1	96.4 $\pm$ 0.1	93.5 $\pm$ 0.3	90.0 $\pm$ 0.8
Qwen 2.5 72b	90.0 $\pm$ 1.8	**84.4** $\pm$ 4.2	**78.2** $\pm$ 7.3	**99.1** $\pm$ 0.0	**98.3** $\pm$ 0.0	**97.2** $\pm$ 0.0
LLaMA 3.1 8b	81.6 $\pm$ 0.5	73.0 $\pm$ 1.3	61.7 $\pm$ 4.8	89.5 $\pm$ 0.1	75.7 $\pm$ 1.6	67.5 $\pm$ 3.0
LLaMA 3.1 70b	82.3 $\pm$ 1.8	68.8 $\pm$ 2.3	30.8 $\pm$ 9.2	87.4 $\pm$ 0.7	80.7 $\pm$ 1.1	71.9 $\pm$ 3.2

Table 4: **Code Equivalence Preservation in AI-Assisted Programming Tasks.** Consistency scores(%) for algorithm transformations at varying abstraction levels (L1-L3) across different architectures and model sizes

Evaluator / Evaluatee	Qwen 2.5 7B			Qwen 2.5 72B		
	L1	L2	L3	L1	L2	L3
Qwen 2.5 7b	86.2 $\pm$ 0.4	78.2 $\pm$ 0.8	72.2 $\pm$ 1.9	85.4 $\pm$ 0.2	80.2 $\pm$ 0.2	71.7 $\pm$ 0.4
Qwen 2.5 72b	87.2 $\pm$ 0.2	80.1 $\pm$ 0.6	79.4 $\pm$ 2.3	**90.8** $\pm$ 0.2	**85.3** $\pm$ 0.5	77.0 $\pm$ 1.9
LLaMA 3.1 8b	84.4 $\pm$ 0.2	72.1 $\pm$ 0.4	61.0 $\pm$ 1.1	88.1 $\pm$ 0.2	76.0 $\pm$ 0.5	60.4 $\pm$ 1.0
LLaMA 3.1 70b	**91.0** $\pm$ 0.5	**85.6** $\pm$ 1.0	**81.5** $\pm$ 2.1	87.9 $\pm$ 0.4	82.7 $\pm$ 0.6	**83.5** $\pm$ 1.0

while the Qwen 2.5 72B evaluator model provides scores of 90.0 at L1, 84.4 at L2, and 78.2 at L3. The consistency is more evident as the path length increases (L1 to L3).

Ablation under evaluatee model. The consistency score of the same evaluated model varies between different evaluator models, as is shown in Table 4 and Table 3. For example, in the AI-assisted programming task, the Qwen 2.5 7B model evaluated by the Qwen 2.5 7B evaluator achieves scores of 86.2 at L1, 78.2 at L2, and 72.2 at L3, while the same model evaluated by the Qwen 2.5 72B evaluator achieves scores of 85.4 at L1, 80.2 at L2, and 71.7 at L3. However, they all share the trend of consistency score decreasing when the path becomes longer.

8 Discussion

BLEU (Papineni et al., 2002) is a traditional metric for evaluating translation similarity. We also measured similarities using BLEU in the same experiments referenced in Tables 1, 2, 5, and 6. For the translation task, we observed strong correlations between embedding similarity and BLEU scores, with Pearson coefficients of 0.859 (L-1), 0.743 (L-2), and 0.716 (L-3). The Spearman coefficients were even higher, at 0.905 (L-1), 0.881 (L-2), and 0.881 (L-3), indicating a consistent monotonic relationship. This aligns with expectations, as BLEU is designed for textual similarity, and embedding-based metrics also capture semantic and structural similarities in text.

Table 5: **Cross-lingual Translation BLEU Score Evaluation.** These BLEU scores(%) evaluate LLMs' ability to maintain translation quality across different languages. The evaluated models vary in model families, scale (1.5B-72B) and path length (1-3).

Evaluatee	**L1**	**L2**	**L3**
GPT 4o mini	**86.0** $\pm$ 0.0	**78.6** $\pm$ 0.1	**68.0** $\pm$ 0.2
Qwen 2.5 1.5b	53.4 $\pm$ 0.3	37.6 $\pm$ 0.3	25.8 $\pm$ 0.3
Qwen 2.5 7b	70.5 $\pm$ 0.3	56.0 $\pm$ 0.7	41.9 $\pm$ 0.7
Qwen 2.5 14b	76.3 $\pm$ 0.1	65.1 $\pm$ 0.2	53.8 $\pm$ 0.2
Qwen 2.5 32b	78.4 $\pm$ 0.1	68.2 $\pm$ 0.2	57.3 $\pm$ 0.3
Qwen 2.5 72b	81.7 $\pm$ 0.1	71.2 $\pm$ 0.1	57.4 $\pm$ 0.2
LLaMA 3.1 8b	59.5 $\pm$ 1.2	47.6 $\pm$ 1.1	36.6 $\pm$ 0.9
LLaMA 3.1 70b	63.1 $\pm$ 2.0	57.3 $\pm$ 1.8	46.9 $\pm$ 2.3

For the coding task, the correlations were surprisingly stronger, with Pearson coefficients of 0.981 (L-1), 0.967 (L-2), and 0.989 (L-3), and Spearman coefficients of 0.833 (L-1), 0.976 (L-2), and 1.000 (L-3). This high correlation is unexpected because BLEU is not designed to compare numerical or functional outputs, such as test case results. However, the results suggest that, in this specific context, BLEU is capturing meaningful relationships between the outputs, possibly due to the structured and deterministic nature of the test case results.

9 Case Study

In this section, we present a case study involving two trees with the same root node. The root node contains a Python function that returns a detailed analysis of the integration of artificial in-

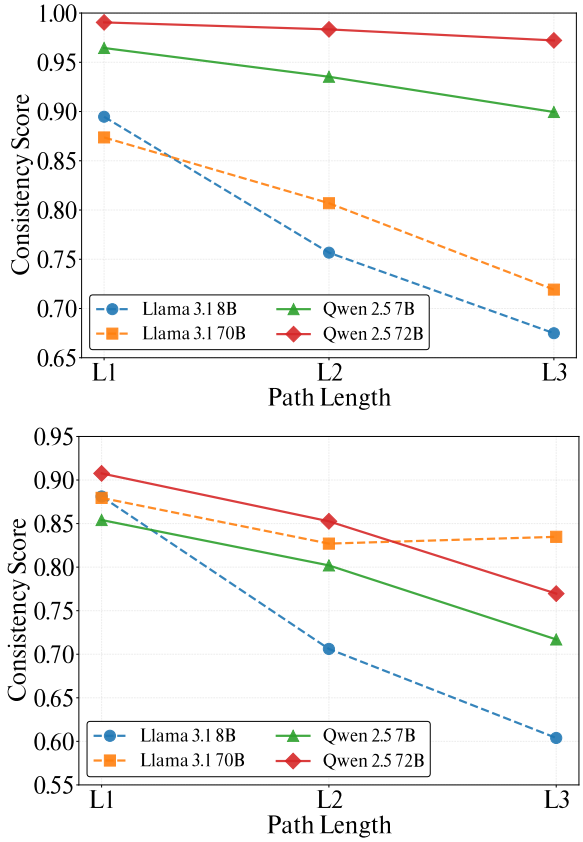

Figure 2: **Consistency scores(%) for various path length**. The upper one shows consistency score on cross-lingual translation and the lower one shows consistency score on code equivalence perservation.

telligence (AI) in healthcare systems. The trees are constructed by applying a series of translation operations to the return value of the ‘main‘ function. The operations include translating the text to French, German, and Spanish, and then translating it back to English. The trees are evaluated to a depth of 3, and we compare the results from two different models: Qwen 2.5 72B and LLaMA 3.1 8B/70B. The code content of these 3 nodes is in Appendix§I. The following operations are applied to the root node:

- Translate the return value of ‘main’ to French, then translate it back to English.
- Translate the return value of ‘main’ to German, then translate it back to English.
- Translate the return value of ‘main’ to Spanish, then translate it back to English.

The first leaf node is generated by the LLaMA 3.1 8B model, while the second leaf node is generated by the LLaMA 3.1 70B model. Both models demonstrate the ability to preserve the core meaning of the original text after multiple translation operations, although slight variations in wording and phrasing are observed. LLaMA 3.1 70B model managed to retain the original text with minimal changes (Levenshtein Distance of **66**), while LLaMA 3.1 8B had introduced more variations (Levenshtein Distance of **215**). This is also consistent with the L-3 consistency scores measured by embedding similarity and BLEU. This case study highlights the robustness of the models in maintaining semantic consistency across iterative transformations.

Table 6: **Code Equivalence BLEU Score Evaluation.** These BLEU scores(%) evaluate LLMs’ ability to maintain code functionality when performing algorithm transformations multiple times. The evaluated models vary in model families, scale and path length.

Evaluatee	L1	L2	L3
GPT 4o mini	78.4 ± 1.1	65.0 ± 2.5	48.3 ± 8.6
Qwen 2.5 1.5b	79.1 ± 0.6	58.0 ± 1.3	23.3 ± 3.7
Qwen 2.5 7b	64.5 ± 0.7	50.3 ± 1.4	31.4 ± 2.4
Qwen 2.5 14b	76.8 ± 0.6	60.4 ± 2.6	51.6 ± 3.8
Qwen 2.5 32b	**79.2** ± 1.1	**71.9** ± 2.9	**64.3** ± 4.3
Qwen 2.5 72b	78.8 ± 1.1	66.5 ± 2.5	48.9 ± 6.4
LLaMA 3.1 8b	69.4 ± 1.1	42.9 ± 2.4	11.7 ± 2.1
LLaMA 3.1 70b	70.5 ± 1.2	58.3 ± 1.9	61.2 ± 3.0

10 Conclusion

In this work, we evaluated the consistency of large language models (LLMs) across translation and AI-assisted programming tasks using the ConsistencyChecker framework. Our results demonstrate that while larger models generally outperform smaller ones, model scale alone does not guarantee superior performance. Specifically, GPT-4o Mini emerges as the state-of-the-art (SOTA) model for translation tasks, achieving the highest consistency scores across all transformation levels. On the other hand, LLaMA 3.1 70B establishes itself as the SOTA model for AI-assisted programming, excelling in generating functionally correct and semantically consistent code. These findings highlight the importance of task-specific optimization and suggest that model scale is not the sole determinant of performance in consistency evaluation.

Limitations

While this study offers valuable insights into the robustness and consistency of large language models (LLMs) through the proposed ConsistencyChecker

framework, it is important to acknowledge its limitations.

The evaluation is restricted to only two tasks—cross-lingual translation and AI-assisted programming. Although these tasks are representative of important LLM capabilities, they do not encompass the full spectrum of potential applications, such as summarization, question answering, or creative writing. Expanding the evaluation to include a broader range of tasks in future work would ensure the generalizability of the findings and provide a more comprehensive assessment of model performance.

Additionally, the study evaluates a limited set of models, primarily focusing on the Qwen and LLaMA families. While these models are widely used and representative of current LLM advancements, they do not cover the full diversity of architectures and training methodologies available in the field. Including other state-of-the-art models, such as GPT, Claude, or PaLM, would offer a more holistic understanding of LLM consistency and robustness across different design choices and training paradigms.

Another limitation lies in the scale and resource constraints of the evaluation. The study examines models ranging from 1.5B to 72B parameters, which, while covering a significant portion of the model scale spectrum, excludes smaller models (*e.g.*, <1B parameters) and extremely large models (*e.g.*, >100B parameters). Furthermore, the computational resources required for generating and evaluating self-consistency trees at scale may limit the feasibility of applying this framework to even larger models or more extensive datasets.

The chosen tasks may also introduce biases based on their inherent difficulty or domain specificity. For instance, cross-lingual translation and code transformation tasks may favor models with specific training data or architectural features, potentially skewing the results. A more diverse set of tasks would help mitigate such biases and provide a more balanced assessment of model performance.

Finally, the study relies solely on automated metrics, such as embedding similarity, BLEU, and ROUGE, to measure consistency. While these metrics provide quantitative measures, they do not fully capture the nuances of human judgment, particularly for tasks requiring high levels of creativity or subjective interpretation. Incorporating human evaluation in future work would enhance the validity and reliability of the results.

Addressing these limitations in future research will strengthen the ConsistencyChecker framework and provide a more comprehensive understanding of LLM consistency and robustness across diverse tasks, models, and evaluation methodologies.

References

Loïc Barrault, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, Shervin Malmasi, Christof Monz, Mathias Müller, Santanu Pal, Matt Post, and Marcos Zampieri. 2019. Findings of the 2019 conference on machine translation (WMT19). In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 1–61, Florence, Italy. Association for Computational Linguistics.

Clark Barrett and Cesare Tinelli. 2018. Satisfiability modulo theories. *Handbook of model checking*, pages 305–343.

Samuel R. Bowman and George E. Dahl. 2021. What will it take to fix benchmarking in natural language understanding? *CoRR*, abs/2104.02145.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *CoRR*, abs/2005.14165.

Chris Callison-Burch, Miles Osborne, and Philipp Koehn. 2006. Re-evaluating the role of Bleu in machine translation research. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 249–256, Trento, Italy. Association for Computational Linguistics.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles

Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Hao Zhang, Banghua Zhu, Michael Jordan, Joseph E. Gonzalez, and Ion Stoica. 2024. Chatbot arena: An open platform for evaluating llms by human preference. *Preprint*, arXiv:2403.04132.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sashank Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2024. Palm: scaling language modeling with pathways. *J. Mach. Learn. Res.*, 24(1).

Edmund M Clarke, Orna Grumberg, and Doron Peled. 1999. *Model checking*. MIT press.

Wendi Cui, Zhuohang Li, Damien Lopez, Kamalika Das, Bradley A. Malin, Sricharan Kumar, and Jiaxin Zhang. 2024. Divide-conquer-reasoning for consistency evaluation and automatic improvement of large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 334–361, Miami, Florida, US. Association for Computational Linguistics.

Irena Gao, Gabriel Ilharco, Scott Lundberg, and Marco Tulio Ribeiro. 2023. Adaptive testing of computer vision models. *Preprint*, arXiv:2212.02774.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard S. Zemel, Wieland Brendel, Matthias Bethge, and Felix A. Wichmann. 2020. Shortcut learning in deep neural networks. *CoRR*, abs/2004.07780.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and Jacob Steinhardt. 2021. Measuring coding challenge competence with APPS. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Juyong Jiang, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim. 2024. A survey on large language models for code generation. *Preprint*, arXiv:2406.00515.

Wai-Chung Kwan, Xingshan Zeng, Yuxin Jiang, Yufei Wang, Liangyou Li, Lifeng Shang, Xin Jiang, Qun Liu, and Kam-Fai Wong. 2024. Mt-eval: A multi-turn capabilities evaluation benchmark for large language models. *Preprint*, arXiv:2401.16745.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Shuai Lu, Daya Guo, Shuo Ren, Junjie Huang, Alexey Svyatkovskiy, Ambrosio Blanco, Colin Clement, Dawn Drain, Daxin Jiang, Duyu Tang, Ge Li, Lidong Zhou, Linjun Shou, Long Zhou, Michele Tufano, MING GONG, Ming Zhou, Nan Duan, Neel Sundaresan, Shao Kun Deng, Shengyu Fu, and Shujie LIU. 2021. CodeXGLUE: A machine learning benchmark dataset for code understanding and generation. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*.

Inbal Magar and Roy Schwartz. 2022. Data contamination: From memorization to exploitation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 157–165, Dublin, Ireland. Association for Computational Linguistics.

Guillermo Marco, Julio Gonzalo, M.Teresa Mateo-Girona, and Ramón Del Castillo Santos. 2024. Pron vs prompt: Can large language models already challenge a world-class fiction author at creative text writing? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 19654–19670, Miami, Florida, USA. Association for Computational Linguistics.

Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. 2024. Large language models: A survey. *Preprint*, arXiv:2402.06196.

Tobias Nipkow, Markus Wenzel, and Lawrence C. Paulson. 2002. *Isabelle/HOL: a proof assistant for higher-order logic*. Springer-Verlag, Berlin, Heidelberg.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Shenbin Qian, Archchana Sindhujan, Minnie Kabra, Diptesh Kanojia, Constantin Orasan, Tharindu Ranasinghe, and Fred Blain. 2024. What do large language models need for machine translation evaluation? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 3660–3674, Miami, Florida, USA. Association for Computational Linguistics.

Matthew Snover, Bonnie Dorr, Rich Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas: Technical Papers*, pages 223–231, Cambridge, Massachusetts, USA. Association for Machine Translation in the Americas.

Menno van Zaanen and Simon Zwarts. 2006. Unsupervised measurement of translation quality using multi-engine, bi-directional translation. In *Proceedings of the 19th Australian Joint Conference on Artificial Intelligence: Advances in Artificial Intelligence*, AI'06, page 1208–1214, Berlin, Heidelberg. Springer-Verlag.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

Jim Waldo and Soline Boussard. 2024. Gpts and hallucination: Why do large language models hallucinate? *Queue*, 22(4):19–33.

Tiannan Wang, Jiamin Chen, Qingrui Jia, Shuai Wang, Ruoyu Fang, Huilin Wang, Zhaowei Gao, Chunzhao Xie, Chuou Xu, Jihong Dai, Yibin Liu, Jialong Wu, Shengwei Ding, Long Li, Zhiwei Huang, Xinle Deng, Teng Yu, Gangan Ma, Han Xiao, Zixin Chen, Danjun Xiang, Yunxia Wang, Yuanyuan Zhu, Yi Xiao, Jing Wang, Yiru Wang, Siran Ding, Jiayang Huang, Jiayi Xu, Yilihamu Tayier, Zhenyu Hu, Yuan Gao, Chengfeng Zheng, Yueshu Ye, Yihang Li, Lei Wan, Xinyue Jiang, Yujie Wang, Siyu Cheng, Zhule Song, Xiangru Tang, Xiaohua Xu, Ningyu Zhang, Huajun Chen, Yuchen Eleanor Jiang, and Wangchunshu Zhou. 2024a. Weaver: Foundation models for creative writing. *Preprint*, arXiv:2401.17268.

Yiming Wang, Pei Zhang, Baosong Yang, Derek F. Wong, and Rui Wang. 2024b. Latent space chain-of-embedding enables output-free llm self-evaluation. *Preprint*, arXiv:2410.13640.

Yue Wang, Hung Le, Akhilesh Deepak Gotmare, Nghi D. Q. Bui, Junnan Li, and Steven C. H. Hoi. 2023. Codet5+: Open code large language models for code understanding and generation. *Preprint*, arXiv:2305.07922.

Warren Weaver. 1952. Translation. In *Proceedings of the Conference on Mechanical Translation*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *CoRR*, abs/2201.11903.

Daniel Yang, Yao-Hung Hubert Tsai, and Makoto Yamada. 2024. On verbalized confidence scores for llms. *Preprint*, arXiv:2412.14737.

Qiang Zhang, Keyang Ding, Tianwen Lyv, Xinda Wang, Qingyu Yin, Yiwen Zhang, Jing Yu, Yuhao Wang, Xiaotong Li, Zhuoyi Xiang, Kehua Feng, Xiang Zhuang, Zeyuan Wang, Ming Qin, Mengyao Zhang, Jinlu Zhang, Jiyu Cui, Tao Huang, Pengju Yan, Renjun Xu, Hongyang Chen, Xiaolin Li, Xiaohui Fan, Huabin Xing, and Huajun Chen. 2024. Scientific large language models: A survey on biological & chemical domains. *Preprint*, arXiv:2401.14656.

Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623. Curran Associates, Inc.

Wenhao Zhu, Hongyi Liu, Qingxiu Dong, Jingjing Xu, Shujian Huang, Lingpeng Kong, Jiajun Chen, and Lei Li. 2024. Multilingual machine translation with large language models: Empirical results and analysis. *Preprint*, arXiv:2304.04675.

A Potential Risks

The potential risks associated with this work include the misuse of the generated artifacts outside of research contexts. To mitigate this, we have included a license in the appendix that specifies the intended use of the artifacts and restricts their use to research purposes only.

B Artifact Use Consistent With Intended Use

We confirm that our use of existing artifacts is consistent with their intended use as specified by their respective licenses. For artifacts created in this work, we explicitly specify their intended use in the documentation, ensuring compatibility with the original access conditions. Derivatives of data accessed for research purposes are restricted to research contexts only.

C Documentation Of Artifacts

We provide comprehensive documentation for the artifacts used and created in this work. This includes details on the domains, languages, and linguistic phenomena covered, as well as the demographic groups represented. This study involves generation of benchmarks and evaluation log files involving English, German, French, and Spanish. Japanese is mentioned in this paper, and is also mentioned in the prompts which guide the generator model to generate operation pairs as benchmark.

D Statistics For Data

We report relevant statistics for the data used and created in this work. This includes the number of examples, details of train/test/dev splits, and other relevant metadata. The four evaluator models involved in this study each generated 2 benchmark files, with one for the translation task and one for the ai-assisted programming task. Each of these benchmark files contains 10 self-consistency root nodes.

E Model Size And Budget

We report the number of parameters in the models used, the total computational budget (*e.g.*, GPU hours), and the computing infrastructure used. Specifically, the models used in this work have the following configurations:

- Qwen 2.5 72B: 72 billion parameters.
- LLaMA 3.1 8B: 8 billion parameters.
- LLaMA 3.1 70B: 70 billion parameters.

This project is conducted on 10 NVIDIA RTX A6000 GPUs. All LLMs with parameter scale less than 8B were hosted locally with vLLM on the A6000s with seed 42. All other LLMs, such as GPT 4o Mini, Qwen 2.5 14B, Qwen 2.5 70B, LLaMA 3.1 72B, are accessed through API. All the LLMs in the LLaMA and Qwen families are instruction tuned versions. All LLMs are not quantized additionally.

F Experimental Setup And Hyperparameters

We discuss the experimental setup, including hyperparameter search and the best-found hyperparameter values. The temperature for text generation was set to 0.6. The tree structure used for evaluation had an out-degree of 3 (operation pairs) and a height of 3. These parameters were chosen based on preliminary experiments to balance diversity and consistency in the generated outputs.

G Parameters For Packages

We used the following packages and reported their implementation, model, and parameter settings:

- `nltk.translate.bleu_score`: For calculating BLEU scores, we used the `sentence_bleu` function with `SmoothingFunction().method1`.

H Information About Use Of AI Assistants

We acknowledge the use of AI assistants in the preparation of this work. The AI assistants were used for generating and refining text, as well as for providing suggestions on experimental design and hyperparameter tuning. All outputs generated by AI assistants were reviewed and validated by the authors to ensure accuracy and relevance.

I Examples of Case Study

We include an example of machine translation consistency evaluation here.

Root Node: def main(): return "In the rapidly evolving landscape of artificial intelligence (AI), the integration of these advanced technologies into healthcare systems represents a significant frontier with profound implications for medical diagnostics and patient care. Recent advancements have demonstrated the potential of AI to enhance diagnostic accuracy, streamline clinical workflows, and personalize treatment plans, thereby improving patient outcomes and reducing healthcare costs. However, the deployment of AI in this sensitive domain is not without its challenges. Ethical considerations, such as data privacy, algorithmic bias, and the potential for job displacement, are paramount. Moreover, the regulatory landscape must adapt to ensure that AI applications meet stringent safety and efficacy standards. Despite these hurdles, the collaboration between tech developers, healthcare providers, and regulatory bodies is fostering a cautious yet optimistic approach to AI integration. The future of healthcare may well be defined by the seamless fusion of human expertise and machine intelligence, creating a hybrid model that leverages the strengths of both to deliver more effective and efficient care."

LLaMA 3.1 8B Leaf Node: def main(): return "In the constantly evolving scenario of artificial intelligence (AI), the integration of these advanced technologies in medical systems represents a significant frontier with important implications for medical diagnosis and patient care. Recent advances have demonstrated the potential of AI to improve diagnostic accuracy, simplify clinical workflows, and personalize treatment plans, leading to better patient outcomes and a reduction in medical costs. However, implementing AI in this sensitive domain does not come without its challenges. Ethical considerations, such as data privacy, algorithmic bias, and job displacement, are fundamental. Furthermore, the regulatory landscape must adapt to ensure that AI applications meet strict standards of safety and efficacy. Despite these difficulties, collaboration between technology developers, healthcare providers, and regulatory bodies is promoting a cautious yet optimistic approach to the integration of AI. The future of healthcare may be defined by the seamless fusion of human experience and artificial intelligence, creating a hybrid model that leverages the strengths of both to offer more effective and efficient care."

LLaMA 3.1 70B Leaf Node: def main(): return "In the rapidly evolving landscape of artificial intelligence (AI), the integration of these advanced technologies into healthcare systems represents a significant frontier with profound implications for medical diagnosis and patient care. Recent advances have demonstrated AI's potential to improve diagnostic accuracy, streamline clinical workflows, and personalize treatment plans, thereby enhancing patient outcomes and reducing healthcare costs. However, the deployment of AI in this sensitive domain is not without challenges. Ethical considerations, such as data privacy, algorithmic bias, and potential job displacement, are paramount. Moreover, the regulatory landscape must adapt to ensure that AI applications meet stringent safety and efficacy standards. Despite these obstacles, collaboration between technology developers, healthcare providers, and regulatory agencies is fostering a cautious yet optimistic approach to AI integration. The future of healthcare may be defined by the seamless fusion of human expertise and machine intelligence, creating a hybrid model that leverages the strengths of both to deliver more effective and efficient care."

Figure 3: **An Example of Root Node code, and Leaf Node Code in a Self-Consistency Tree of Height 3.**