STED and Consistency Scoring: A Framework for Evaluating LLM Structured Output Reliability

Guanghui Wang[1*] **Jinze Yu**[1†] **Xing Zhang**[1] **Dayuan Jiang**[1] **Yin Song**[2]
Tomal Deb[1] **Xuefeng Liu**[1] **Peiyang He**[1]
[1]AWS Generative AI Innovation Center
[2]AWS WWSO SA Field Initiatives

Abstract

Large Language Models (LLMs) are increasingly deployed for structured data generation, yet output consistency remains critical for production applications. We introduce a comprehensive framework for evaluating and improving consistency in LLM-generated structured outputs. Our approach combines: (1) STED (Semantic Tree Edit Distance), a novel similarity metric balancing semantic flexibility with structural strictness when comparing JSON outputs, and (2) a consistency scoring framework aggregating multiple STED measurements across repeated generations to quantify reliability. Through systematic experiments on synthetic datasets with controlled schema, expression, and semantic variations, we demonstrate STED achieves superior performance ($0.86 - 0.90$ similarity for semantic equivalents, 0.0 for structural breaks) compared to existing metrics including TED, BERTScore, and DeepDiff. Applying our framework to benchmark six LLMs reveals significant variations: Claude-3.7-Sonnet demonstrates exceptional consistency, maintaining near-perfect structural reliability even at high temperatures ($T = 0.9$), while models like Claude-3-Haiku and Nova-Pro exhibit substantial degradation requiring careful tuning. Our framework enables practical applications including targeted model selection for structured tasks, iterative prompt refinement for reproducible results, and diagnostic analysis to identify inconsistency root causes. This work provides theoretical foundations and practical tools for ensuring reliable structured output generation in LLM-based production systems.

1 Introduction

Large Language Models (LLMs) have become integral to production systems requiring structured data generation, particularly in JSON format for APIs, data extraction pipelines, and automated workflows. However, evaluating the consistency and quality of LLM-generated structured outputs presents unique challenges that existing evaluation methods fail to adequately address.

A fundamental issue in evaluating structured outputs is the mismatch between evaluation methods and structured semantics. Consider two structured objects with identical content but different key ordering—while the structured specification treats these as equivalent, popular evaluation methods like BERTScore [1] assign them significantly different similarity scores due to position-sensitive embeddings, leading to false negatives in production systems.

The challenge extends beyond simple key reordering. LLMs frequently generate functionally equivalent structures with various forms of semantic equivalence: naming convention differences

[*]Equal contribution
[†]Corresponding author

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Structured Probabilistic Inference and Generative Modeling.

("user_name" versus "userName"), array reorderings where sequence is not semantically significant, structural reorganizations preserving information content, and type representation variations ("true" versus true, "123" versus 123). Each represents valid structured data conveying identical information, yet existing metrics fail to recognize their equivalence.

Current evaluation approaches exhibit systematic limitations when applied to structured outputs. BERTScore [1], while effective for natural language evaluation, suffers from order sensitivity that violates JSON's order-agnostic semantics for object properties, systematically underestimating similarity for reordered but identical structures. Tree Edit Distance (TED) methods focus purely on structural differences without considering semantic equivalence, while exact matching approaches are overly restrictive for practical applications.

DeepDiff [2] and similar structural comparison tools can ignore ordering through configuration but lack semantic understanding. They treat "email" and "email_address" as completely different keys, missing obvious semantic relationships that humans would recognize.

These limitations have significant practical consequences. In production systems processing thousands of API responses, false negatives from order sensitivity can trigger unnecessary alerts, increase operational overhead, and mask genuine inconsistencies.

While traditional edit distances like Tree Edit Distance (TED) have been applied to structured data comparison, they fail to handle the semantic equivalences common in LLM outputs. Recent advances in learned edit distances [3] improve classification accuracy but focus on discriminative tasks rather than consistency evaluation. We propose STED (Semantic Tree Edit Distance), which adapts edit distance specifically for LLM output evaluation, balancing semantic equivalence with structural validity.

Unlike embedding-based approaches that learn vectorial representations [3], STED directly incorporates JSON semantics through three key innovations: (1) **Semantic-Enhanced Tree Edit Distance** that recognizes semantically equivalent keys and values while preserving structural constraints, (2) **Order-Invariant Matching** using Hungarian algorithm for optimal element pairing, and (3) **Multi-Level Similarity** integrating structural, key, value, and type similarities with configurable weights.

Our experiments demonstrate STED's superior discrimination: while BERTScore and DeepDiff score > 0.95 across all variations—failing to distinguish schema violations from benign reorderings—STED correctly identifies critical differences (0.23 for schema violations vs > 0.95 for order variations), achieving $4\times$ better discrimination.

As LLMs increasingly power production systems, STED provides a theoretically grounded solution for evaluating structured output consistency, enabling reliable deployment through accurate distinction between critical errors and acceptable variations.

2 Related Work

2.1 Edit Distance and Structured Data Comparison

The tree edit distance problem, formalized by Tai [4], provides the foundation for comparing hierarchical structures. Zhang and Shasha [5] proposed the first polynomial-time algorithm, later improved by RTED [6] and extended with move operations [7]. However, these classical approaches rely on exact node matching without semantic understanding. Recent learning-based approaches [3, 8] explore adaptive embeddings, while graph edit distance methods [9, 10] handle complex structures, inspiring our semantic enhancements. JSON-specific tools like RFC 6902 [11] and DeepDiff [2] provide structural comparison with optional order-insensitive matching but lack semantic awareness. JSON Schema validation [12] focuses on conformance rather than similarity, while query languages like JMESPath [13] and JSONPath [14] target extraction, not comparison.

2.2 Neural Approaches to Similarity

BERTScore [1] revolutionized text evaluation using contextualized embeddings from BERT [15], but its position sensitivity makes it unsuitable for JSON where key ordering is irrelevant. Advances like Sentence-BERT [16], SimCSE [17], and Universal Sentence Encoder [18] improve embedding quality but process JSON as flat strings, losing structural information. Code similarity work including CodeBERT [19] and GraphCodeBERT [20] demonstrates the importance of combining structural and

semantic features. Graph neural networks offer another perspective through GraphSAGE [21], GAT [22], Tree-LSTM [23], and GraphFormers [24]. While powerful, these require training data and are computationally expensive compared to our approach.

2.3 LLM Structured Output Generation and Consistency

Recent work addresses structured output challenges in LLMs. Bubeck et al. [25] analyze GPT-4's structured generation capabilities, while HELM [26] provides holistic evaluation focusing on task performance. SLOT [27] and StructuredRAG [28] address output structuring, with StructEval [29] benchmarking generation capabilities. Format control studies [30, 31] examine verification and schema conformance. LLM consistency research spans multiple dimensions. Elazar et al. [32] and Raj et al. [33] examine consistency across paraphrases, while recent studies investigate stability [34], non-determinism [35], and automated analysis [36, 37]. Temperature effects [38, 39] and prompt stability [40] are also explored, but these focus on input variations rather than output consistency for identical inputs.

2.4 Semantic Matching and Optimal Assignment

Semantic similarity in structured data draws from ontology matching [41] and schema matching [42], with approaches like SimFlood [43] and COMA [44] combining multiple strategies. Neural methods including DeepMatcher [45] and SMAT [46] achieve high accuracy but target one-time alignment rather than continuous evaluation. Distance functions for structured data [47, 48, 49] provide theoretical foundations. The Hungarian algorithm [50, 51] enables polynomial-time optimal assignment, extended recently for large-scale approximations [52]. TreeKernel [53] and subsequent work [54] apply assignment to tree matching, inspiring our combination with semantic similarity for arrays and keys in JSON.

Despite extensive related work, existing methods fail to address structured LLM-generated outputs consistency evaluation challenges: (1) Methods focus on either structure or semantics but not both; (2) Tools are either order-agnostic or overly sensitive; (3) No method handles LLM-specific variation patterns. STED bridges these gaps through unified structural-semantic analysis with appropriate order handling and granular insights.

3 Methodology

We present STED (Semantic Tree Edit Distance), a novel framework for evaluating consistency in LLM-generated structured outputs. While traditional metrics treat structured formats like JSON, XML, and HTML as flat text, we recognize their hierarchical nature by transforming them into tree representations. This allows us to reframe the structured output consistency problem as tree consistency evaluation. Our approach extends classical tree edit distance algorithms with semantic similarity capabilities to compute pairwise distances, then aggregates these distances across multiple LLM generations into a normalized consistency score. This two-stage process addresses the fundamental challenge of quantifying output reliability when LLMs produce functionally equivalent but syntactically different structures.

3.1 Problem Formulation

Let $\mathcal{O} = \{o_1, o_2, ..., o_n\}$ denote structured outputs generated by an LLM for identical inputs. While our framework applies to any hierarchical format (JSON, XML, HTML), we use JSON for clarity. Each output o_i is represented as a tree $T_i = (V_i, E_i)$, where V_i contains nodes (keys, values, structural elements) and E_i encodes parent-child relationships.

The consistency evaluation problem involves two stages:

Pairwise Similarity: Given trees T_1 and T_2, compute a similarity score $s(T_1, T_2) \in [0, 1]$ that captures semantic and structural equivalence, where $s = 1$ indicates perfect consistency and $s = 0$ indicates fundamental incompatibility.

Consistency Score: Given n outputs, aggregate pairwise similarities into a global consistency score $C(\mathcal{O}) \in [0, 1]$ that quantifies the LLM's reliability:

$$C(\mathcal{O}) = \frac{2}{n(n-1)} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} s(T_i, T_j) \tag{1}$$

The core challenge is distinguishing benign variations from critical differences that affect functionality, while providing an interpretable consistency metric for production deployment.

3.2 Tree Representation of Structured Data

We transform structured documents into trees $T = (V, E)$ where each node $v \in V$ contains: **Type**: $\{\text{object}, \text{array}, \text{string}, \text{number}, \text{boolean}, \text{null}\}$; **Label**: Key name for object properties; **Value**: Raw value for primitives, child list for arrays; **Path**: Hierarchical path for unique identification.

The transformation rules are: (1) **Objects** become internal nodes with labeled edges to child values, (2) **Arrays** remain as nodes containing element lists, recursively transformed if complex, and (3) **Primitives** map to leaf nodes with actual values. This preserves array structure while enabling order-invariant matching and maintaining the semantic distinction between arrays and objects.

3.3 Semantic Tree Edit Distance

Our STED algorithm extends classical tree edit distance with semantic awareness through three operations: **Insert** (cost $\gamma_{ins}(v)$), **Delete** (cost $\gamma_{del}(v)$), and **Update** (cost $\gamma_{upd}(v_1, v_2)$). The semantic update cost combines multiple dimensions:

$$\gamma_{upd}(v_1, v_2) = w_s \cdot \gamma_{struct}(v_1, v_2) + w_c \cdot \gamma_{content}(v_1, v_2) \tag{2}$$

where γ_{struct} measures structural similarity using embedding-based comparison, and $\gamma_{content}$ evaluates value similarity with type-aware costs. For arrays and objects, we employ Hungarian algorithm for optimal child matching, ensuring order-invariant comparison.

3.4 Semantic Similarity Computation

Field names are normalized (e.g., `"userName"` $\rightarrow$ `"user name"`) to capture semantic relationships. We compute similarity using embeddings with cosine similarity for texts < 300 characters; longer texts are chunked recursively with 50-character overlaps. We used Amazon Titan Text Embeddings v2 in our implementation, though the choice of embedding model has minimal impact on STED's performance since we primarily match short field names and values where most modern embedding models achieve similar semantic understanding. The framework remains model-agnostic and can use any embedding model (e.g., all-MiniLM-L6-v2, Sentence-BERT, OpenAI embeddings). Type-aware comparison ensures appropriate metrics: semantic for strings, exact for numbers, order-invariant for arrays. This recognizes functional equivalence (e.g., {`"user_name": "John"`} $\equiv$ {`"userName": "John"`}) while detecting type violations that break compatibility.

3.5 Optimal Subtree Matching

Traditional tree edit distance algorithms process nodes sequentially, potentially missing globally optimal alignments. We address this through Hungarian algorithm-based matching at each tree level.

For trees T_1 and T_2 with children sets C_1 and C_2, we formulate optimal matching as an assignment problem:

$$\text{OptimalCost}(T_1, T_2) = \min_{\pi} \left[\sum_{(i,j) \in \pi} d(c_1^i, c_2^j) + \sum_{i \notin \pi} \gamma_{del}(c_1^i) + \sum_{j \notin \pi} \gamma_{ins}(c_2^j) \right] \tag{3}$$

where π represents the matching between children, $d(\cdot, \cdot)$ is the recursive distance, and unmatched nodes incur insertion/deletion costs.

The algorithm constructs a cost matrix $M \in \mathbb{R}^{\max(|C_1|,|C_2|) \times \max(|C_1|,|C_2|)}$ where M_{ij} represents the cost of matching child i from T_1 with child j from T_2. The Hungarian algorithm finds the minimum-cost assignment in $O(n^3)$ time, ensuring globally optimal child alignment rather than greedy local decisions.

3.6 Similarity Score Normalization

The STED algorithm normalizes the computed distance at each tree level to produce an interpretable similarity score in [0,1]:

$$\text{STED}(T_1, T_2) = 1 - \min\left(1, \frac{d_{\text{matched}} + \lambda \cdot \Delta_{\text{unmatched}}}{\max(|C_1|, |C_2|)}\right) \tag{4}$$

where d_{matched} is the total cost from Hungarian assignment, $\Delta_{\text{unmatched}} = ||C_1| - |C_2||$ penalizes size differences with weight $\lambda = 0.1$. This per-level normalization ensures consistent scoring regardless of tree depth.

3.7 Consistency Score Calculation

Let $\{s_i\}_{i=1}^{n}$ be the similarity values between responses across different runs, with empirical standard deviation $\sigma = \text{std}(s_1, \ldots, s_n)$. To normalize for scale, we compute the maximum possible standard deviation for n values in $[0, 1]$:

$$\sigma_{\max} = \text{std}\big(\underbrace{0, \ldots, 0}_{\lfloor n/2 \rfloor}, \underbrace{1, \ldots, 1}_{\lceil n/2 \rceil}\big).$$

We then define the normalized deviation

$$\hat{\sigma} = \begin{cases} \dfrac{\sigma}{\sigma_{\max}}, & \sigma_{\max} > 0, \\ 0, & \text{otherwise.} \end{cases}$$

Finally, the consistency score is given by

$$\text{ConsistencyScore}(s_1, \ldots, s_n) = \left(\frac{1}{1 + 2\hat{\sigma}}\right)^{\alpha},$$

where $\alpha = 20$ is a steepness factor that amplifies the typically small deviations observed in model outputs, providing better discrimination in the common low-deviation range while saturating for rare large deviations. If $n \leq 1$, we set ConsistencyScore $= 1$.

3.8 Computational Complexity

STED requires $O(N \times B^3)$ time where N is total node count and B is maximum branching factor, with the Hungarian algorithm contributing the cubic factor. Space complexity is $O(B^2 + D)$ for cost matrices and recursion depth D. Large arrays/objects with high branching factor B dominate cost, while tree depth affects only space linearly, making STED tractable for typical structured outputs but potentially expensive for highly-branched structures.

4 Experiments and Results

Our experimental evaluation comprises two complementary studies. First, we validate our method's effectiveness using synthetic datasets with controlled variations to verify its ability to accurately quantify similarity degradation. Second, we deploy our framework to benchmark the structured output consistency of six LLMs available through Amazon Bedrock: Claude 3.7 Sonnet, Claude 3.5 Sonnet V2, Claude 3.5 Haiku, Claude 3 Haiku, Llama-3.3-70B, and Nova Pro. This model selection represents diverse architectures and optimization strategies—from efficiency-focused (Haiku variants) to performance-focused (Sonnet variants). Our goal is validating the evaluation framework's effectiveness rather than exhaustive model coverage. The chosen models sufficiently demonstrate our method's ability to reveal consistency patterns (temperature sensitivity, structural-semantic gaps, degradation profiles) that represent fundamental behaviors in autoregressive language models. The framework itself is model-agnostic and can evaluate any LLM with structured output capabilities.

4.1 Method Effectiveness Verification

4.1.1 Variation Taxonomy

LLM-generated structured data exhibits three distinct variation types: **Schema Variation**: Structural modifications including field name changes, structure flattening, and hierarchy alterations that affect parseability; **Expression Variation**: Lexical modifications preserving semantic meaning through synonym substitution, paraphrasing, and abbreviation usage; **Semantic Variation**: Fundamental content changes that alter data meaning, potentially causing incorrect interpretations.

4.1.2 Dataset Construction

Our evaluation leverages 2,400 synthetic test cases systematically generated from a diverse set of 80 base samples. We begin with 80 ShareGPT-formatted JSON samples from Quiz Generation and Structured Output datasets (75 valid after parsing errors), which serve as seeds for controlled variation generation. These base samples ensure structural diversity: depths range from 2–7 levels (mean 4.0±1.0, mode 4), field counts span 4–228 (mean 42.3±37.8), and include realistic type distributions (68.2% strings, 18.4% integers, 13.4% complex types). A comprehensive distribution analysis is provided in Appendix A.

Gradual Variations (2,250 samples): We create variants with controlled modification ratios from 0.1 to 1.0 (10 levels) for: (1) *Field Name Variants* - semantically equivalent keys (e.g., "user_name" $\rightarrow$ "userName"), (2) *Expression Variants* - paraphrased values preserving meaning, and (3) *Semantic Variants* - content changes affecting functionality. Each category yields 750 samples (75 $\times$ 10 levels).

Structural Variations (150 samples): We generate single variants for: (1) *Flattened Structure* - nested-to-flat transformations potentially causing field collisions, and (2) *Nested Changes* - modified hierarchies breaking API compatibility. These binary changes yield 75 samples each.

4.1.3 Baseline Methods

We compare STED against three established approaches: **TED**: Tree Edit Distance using Zhang-Shasha algorithm with default cost functions; **BERTScore**: Adapted for JSON by serializing structures with sorted keys, computing F1 scores from BERT embedding alignments; **DeepDiff**: Rule-based structural comparison using Deep Distance metric, converted to similarity as $1 - \text{DeepDistance}$.

4.2 LLM Consistency Benchmarking

We design a comprehensive framework to evaluate LLM consistency in structured output generation. For each dataset sample, we augment prompts with extracted ground truth schemas. Our protocol executes each of 75 samples 10 times per temperature (0.1-0.9), yielding **750 outputs per temperature setting** and 6,750 total outputs per model for robust consistency analysis.

We employ three evaluation modes: (1) structural consistency measuring format adherence, (2) semantic consistency evaluating content preservation, and (3) hybrid approach with equal weighting ($\alpha = 0.5$). This multi-faceted evaluation reveals how temperature variations affect different aspects of structured output generation.

4.3 Method Effectiveness Verification

4.3.1 Schema Variation Analysis

Figure 1 reveals how methods handle structural variations in LLM outputs. For field name variations using semantic equivalents, STED maintains stable similarity across all variation ratios, correctly recognizing functional equivalence. TED degrades significantly due to lack of semantic understanding, while BERTScore's excessive permissiveness may mask important structural differences. For structural modifications (flattening and nesting changes), STED correctly assigns zero similarity, recognizing these as breaking changes for downstream systems. Other methods problematically tolerate these changes with non-zero scores that could allow incompatible outputs to pass consistency checks.

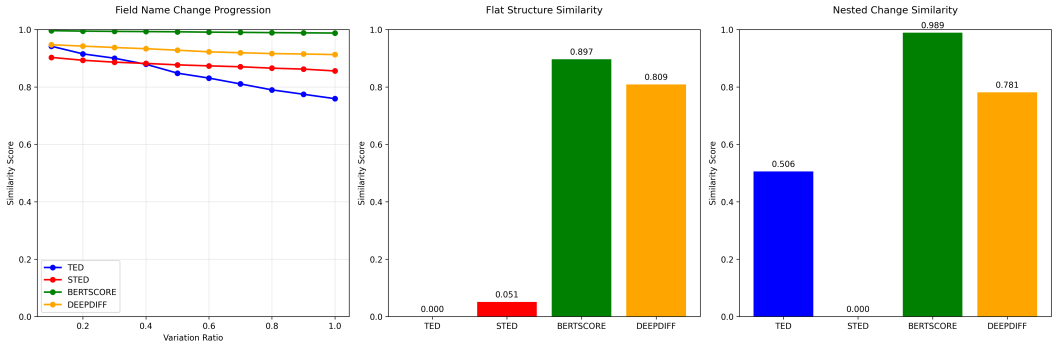

Figure 1: Similarity scores across schema variation types for four consistency evaluation methods. Field name changes (0.1-1.0 ratios) show gradual similarity degradation, with BERTScore maintaining highest scores. For structural variations, STED achieves zero similarity on nested changes but moderate scores on flat structures. All methods use similarity scores ranging from 0 to 1.

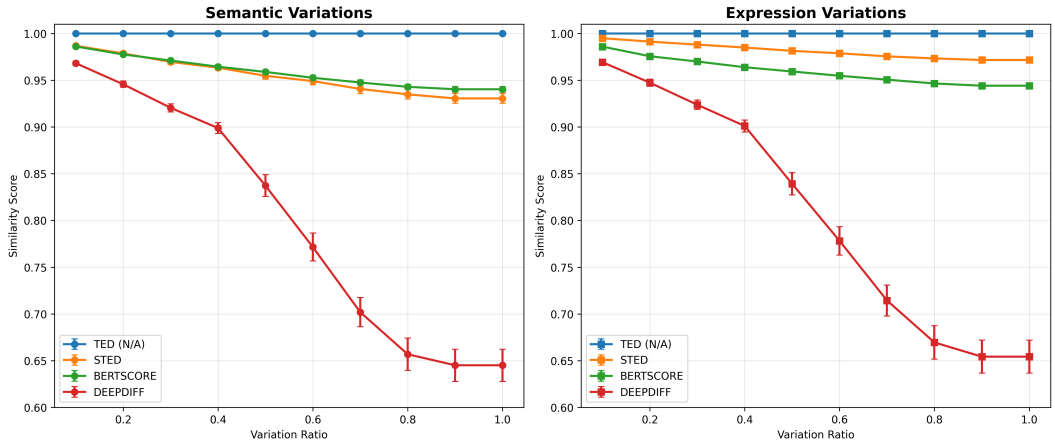

Figure 2: Similarity score progression under semantic variations (left) and expression variations (right). Ideal behavior: sensitivity to semantic changes, robustness to expression changes. STED achieves optimal balance with controlled degradation for semantic variations while maintaining high scores for expression variations, aligning with human consistency perception. (TED remains constant at 1.0 as it measures only structural similarity, not content.)

4.3.2 Content Variation Analysis

Figure 2 evaluates content variation handling, critical for applications like content moderation where semantic accuracy impacts outcomes. For semantic variations, STED (0.954±0.039) and BERTScore (0.958±0.025) show statistically equivalent calibrated sensitivity ($p=0.600$), while DeepDiff over-reacts (0.799±0.165) and TED remains insensitive (1.0, structure-only). For expression variations, STED achieves superior robustness (0.981±0.017, $p<0.001$), correctly preserving high similarity for paraphrases, unlike DeepDiff which incorrectly penalizes valid rewording (0.805±0.164).[3] STED's differential response—5.2% degradation for semantic changes versus 1.9% for expression changes—enables threshold-based detection of genuine semantic drift while accepting natural language variation, providing human-aligned consistency assessment for practical applications.

4.4 Model Consistency Analysis

Figure 3 reveals critical insights into consistency patterns across six language models, providing guidance for model selection and parameter tuning.

[3]Detailed statistical analysis in Supplementary Section B.

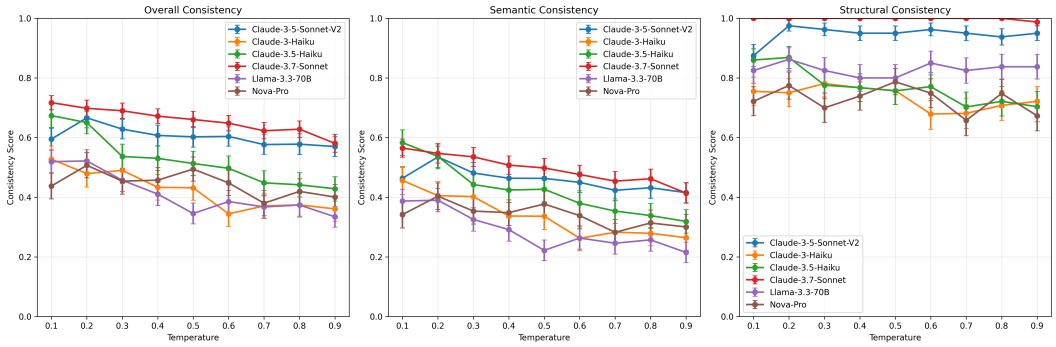

Figure 3: Model consistency across temperature settings. Higher scores indicate better consistency between outputs for the same prompts. Error bars show standard deviation across 75 test cases per temperature. *Detailed numerical results in Supplementary Tables 12–15.*

4.4.1 Temperature Response Patterns

All models exhibit inverse temperature-consistency relationships, validating our evaluation framework. Claude-3.7-Sonnet demonstrates superior stability with only 19% degradation from T=0.1 to T=0.9, maintaining 0.658±0.240 mean consistency. Claude-3.5-Haiku shows highest sensitivity (46% decline), while Claude-3-5-Sonnet-V2 plateaus between T=0.2-0.6, requiring higher temperatures for diversity.

4.4.2 Structural Preservation Under Diversity

Claude-3.7-Sonnet achieves near-perfect structural consistency (0.999±0.037) across all temperatures, demonstrating exceptional format preservation even at T=0.9. This 0.5-point structural-semantic gap indicates sophisticated behavior: varying content while preserving format—ideal for template-based generation. Claude-3-5-Sonnet-V2 similarly maintains 0.946±0.227 structural consistency while semantic scores vary more widely (0.459±0.318), confirming models can reliably produce diverse content within consistent formats.

4.4.3 Deployment Considerations

The evaluation reveals key selection criteria. Structural consistency ranges from near-perfect (Claude-3.7-Sonnet) to moderate (Nova-Pro: 0.728±0.430). Temperature responses vary from smooth degradation to erratic patterns (Nova-Pro peaks at T=0.5). For production: T=0.1-0.3 maximizes consistency, T=0.4-0.6 balances diversity with acceptable degradation, $T \geq 0.7$ causes significant losses (>30% semantic drop). The structural-semantic gap guides task selection: larger gaps suit template-based generation, smaller gaps indicate uniform variation for free-form tasks.

5 Conclusion

We present STED and consistency score, a novel metric for evaluating consistency in LLM-generated structured outputs that balances semantic flexibility with structural strictness. Through controlled experiments on synthetic datasets, we demonstrate STED's superiority: it correctly assigns zero similarity to structure-breaking modifications while maintaining robustness to semantically equivalent variations (0.86-0.90), outperforming TED, BERTScore, and DeepDiff. Our benchmarking of six state-of-the-art LLMs reveals critical insights for production deployment. The proposed consistency score enables three key applications: (1) model selection for structured output tasks, providing targeted evaluation beyond general-purpose benchmarks; (2) prompt refinement through iterative optimization, enabling developers to craft prompts yielding reproducible outputs; and (3) diagnostic analysis to identify factors contributing to inconsistency. These capabilities make our framework a practical tool for improving the reliability of LLM-based systems in production environments where structured output consistency is paramount. Our work has several limitations. We focus exclusively on JSON format and evaluate only six models with 75 samples, potentially missing patterns in other

structured formats or newer models. The semantic similarity metrics may not capture domain-specific equivalences, and the computational cost of multiple generations for consistency evaluation may limit adoption. Future work should extend to other formats, expand model coverage, and optimize evaluation efficiency.

References

[1] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. In *ICLR*, 2020.

[2] Sep Dehpour et al. Deepdiff: Deep difference and search of any python object/data, 2025.

[3] Benjamin Paaßen, Claudio Gallicchio, Alessio Micheli, and Barbara Hammer. Tree edit distance learning via adaptive symbol embeddings, 2018. arXiv:1806.05009v3 [cs.LG] 16 Jul 2018.

[4] Kuo-Chung Tai. The tree-to-tree correction problem. *Journal of the ACM*, 26(3):422–433, 1979.

[5] Kaizhong Zhang and Dennis Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing*, 18(6):1245–1262, 1989.

[6] Mateusz Pawlik and Nikolaus Augsten. Rted: a robust algorithm for the tree edit distance. *Proceedings of the VLDB Endowment*, 5(4):334–345, 2011.

[7] Sudarshan S Chawathe, Anand Rajaraman, Hector Garcia-Molina, and Jennifer Widom. Change detection in hierarchically structured information. In *ACM SIGMOD Record*, volume 25, pages 493–504, 1996.

[8] Benjamin Paaßen. Revisiting the tree edit distance and its backtracing: A tutorial, 2022. arXiv:1805.06869v4 [cs.DS] 14 Sep 2022.

[9] Eeshaan Jain, Indradyumna Roy, Saswat Meher, Soumen Chakrabarti, and Abir De. Graph edit distance with general costs using neural set divergence, 2024. arXiv:2409.17687v2 [cs.ML] 4 Nov 2024.

[10] Chengzhi Piao, Tingyang Xu, Xiangguo Sun, Yu Rong, Kangfei Zhao, and Hong Cheng. Computing graph edit distance via neural graph matching. *Proceedings of the VLDB Endowment*, 16(8):1817–1829, 2023. PVLDB Reference Format.

[11] P Bryan, K Zyp, and M Nottingham. Javascript object notation (json) patch, 2013.

[12] Austin Wright, Henry Andrews, Ben Hutton, and Greg Dennis. Json schema: A media type for describing json documents, 2022.

[13] James Saryerwinnie. Jmespath: A query language for json, 2014.

[14] Stefan Goessner. Jsonpath - xpath for json, 2007.

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186, 2019.

[16] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *EMNLP*, pages 3982–3992, 2019.

[17] Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. In *EMNLP*, pages 6894–6910, 2021.

[18] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, et al. Universal sentence encoder for english. In *EMNLP*, pages 169–174, 2018.

[19] Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, et al. Codebert: A pre-trained model for programming and natural languages. In *EMNLP Findings*, pages 1536–1547, 2020.

[20] Daya Guo, Shuo Ren, Shuai Lu, Zhangyin Feng, Duyu Tang, et al. Graphcodebert: Pre-training code representations with data flow. In *ICLR*, 2021.

[21] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, pages 1024–1034, 2017.

[22] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018.

[23] Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. In *ACL*, pages 1556–1566, 2015.

[24] Junhan Yang, Zheng Liu, Shitao Xiao, Chaozhuo Li, Defu Lian, et al. Graphformers: Gnn-nested transformers for representation learning on textual graph. In *NeurIPS*, pages 2583–2595, 2021.

[25] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

[26] Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yuhui Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*, 2022.

[27] Darren Yow-Bang Wang, Zhengyuan Shen, Soumya Smruti Mishra, Zhichao Xu, Yifei Teng, and Haibo Ding. Slot: Structuring the output of large language models. *arXiv preprint arXiv:2405.04016*, 2024.

[28] Connor Shorten, Charles Pierse, Thomas Benjamin Smith, Erika Cardenas, Akanksha Sharma, John Trengrove, and Bob van Luijt. Structuredrag: Json response formatting with large language models, 2024. arXiv:2408.11061v1 [cs.CL] 7 Aug 2024.

[29] Jialin Yang, Dongfu Jiang, Lipeng He, Sherman Siu, Yuxuan Zhang, Disen Liao, Zhuofeng Li, Huaye Zeng, Yiming Jia, Haozhe Wang, Benjamin Schneider, Chi Ruan, Wentao Ma, Zhengbao Lyu, Yifei Wang, Yi Lu, Quy Duc Do, Ziyan Jiang, Ping Nie, and Wenhu Chen. StructEval: Benchmarking LLMs' capabilities to generate structural outputs. *arXiv preprint arXiv:2505.20139*, 2025.

[30] Zhaoyang Wang, Jinqi Jiang, Huichi Zhou, Wenhao Zheng, Xuchao Zhang, Chetan Bansal, and Huaxiu Yao. Verifiable format control for large language model generations. In *Findings of the Association for Computational Linguistics: NAACL 2025*, 2025.

[31] Saibo Geng, Hudson Cooper, Michał Moskal, Samuel Jenkins, Julian Berman, Nathan Ranchin, Robert West, Eric Horvitz, and Harsha Nori. Generating structured outputs from language models: Benchmark and studies. *arXiv preprint arXiv:2501.10868*, 2025. Also known as JSONSchemaBench.

[32] Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Schütze, and Yoav Goldberg. Measuring and improving consistency in pretrained language models. *Transactions of the Association for Computational Linguistics*, 9:1012–1031, 2021.

[33] Harsh Raj, Domenic Rosati, and Subhabrata Majumdar. Measuring reliability of large language models through semantic consistency. *arXiv preprint arXiv:2211.05853*, 2023.

[34] Berk Atil, Alexa Chittams, Liseng Fu, Ferhan Ture, Lixinyu Xu, and Breck Baldwin. Llm stability: A detailed analysis with some surprises. *arXiv preprint arXiv:2408.04667*, 2024. v2.

[35] Breck Baldwin, Javier Rando, and Florian Tramèr. Non-determinism of "deterministic" LLM settings. *arXiv preprint arXiv:2408.04667*, 2024. Version v5.

[36] Aditya Patwardhan, Vivek Vaidya, and Ashish Kundu. Automated consistency analysis of llms. *arXiv preprint arXiv:2502.07036*, 2025. v1.

[37] Xiaoyuan Wu, Weiran Lin, Omer Akgul, and Lujo Bauer. Estimating llm consistency: A user baseline vs surrogate metrics. *arXiv preprint arXiv:2505.23799*, 2025.

[38] Matthew Renze and Erhan Guven. The effect of sampling temperature on problem solving in large language models, 2024. arXiv:2402.05201v3 [cs.CL] 2 Oct 2024.

[39] Matthew Renze and Erhan Guven. The effect of sampling temperature on problem solving in large language models. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 7346–7356, Miami, Florida, USA, November 2024. Association for Computational Linguistics.

[40] Ke Chen et al. Prompt stability matters: Evaluating and optimizing auto-generated prompt in general-purpose systems. *arXiv preprint arXiv:2505.13546*, 2025.

[41] Jérôme Euzenat and Pavel Shvaiko. *Ontology matching*. Springer, 2013.

[42] Erhard Rahm and Philip A Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350, 2001.

[43] Sergey Melnik, Hector Garcia-Molina, and Erhard Rahm. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *ICDE*, pages 117–128, 2002.

[44] Hong-Hai Do and Erhard Rahm. Coma: a system for flexible combination of schema matching approaches. In *VLDB*, pages 610–621, 2002.

[45] Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, et al. Deep learning for entity matching: A design space exploration. In *SIGMOD*, pages 19–34, 2018.

[46] Chen Zhang, Yuxiang Li, Nan Du, Wei Fan, and Philip S Yu. Smat: An attention-based deep learning framework for semantic matching. In *WWW*, pages 3365–3375, 2021.

[47] Santiago Ontañón. An overview of distance and similarity functions for structured data. *arXiv preprint arXiv:2002.07420*, 2020.

[48] Hans-Peter Kriegel and Stefan Schönauer. Similarity search in structured data. *University of Munich, Institute for Computer Science*, 2005. Database Systems Group.

[49] Rui Yang, Panos Kalnis, and Anthony K. H. Tung. Similarity evaluation on tree-structured data. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, SIGMOD '05, pages 754–765, Baltimore, Maryland, USA, June 2005. ACM.

[50] Harold W Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

[51] James Munkres. Algorithms for the assignment and transportation problems. *Journal of the SIAM*, 5(1):32–38, 1957.

[52] Chris Schwiegelshohn and Uwe Schwiegelshohn. A learned approach to the online min-sum set cover problem. In *ICLR*, 2022.

[53] Michael Collins and Nigel Duffy. Convolution kernels for natural language. In *NeurIPS*, pages 625–632, 2001.

[54] Alessandro Moschitti. Efficient convolution kernels for dependency and constituent syntactic trees. *ECML*, pages 318–329, 2006.

A Dataset Characteristics and Distribution Analysis

We provide a comprehensive analysis of the 75 base samples used to generate our 2,400 synthetic test cases. This analysis demonstrates the representativeness and diversity of our dataset across multiple dimensions of JSON structural complexity.

A.1 Structural Complexity Distribution

Table 1 shows the distribution of JSON depths across our base samples, with the majority (58.7%) at depth 4, aligning with typical enterprise API structures.

Table 1: JSON Depth Distribution Across Base Samples

Depth Level	Count	Percentage	Interpretation
Depth 2	8	10.7%	Simple, flat structures
Depth 3	7	9.3%	Moderately nested
Depth 4	**44**	**58.7%**	**Most common depth**
Depth 5	13	17.3%	Complex nested structures
Depth 6	2	2.7%	Highly complex
Depth 7	1	1.3%	Maximum complexity

Table 2 presents the field count distribution, demonstrating coverage from simple configurations to complex documents.

Table 2: Field Count Distribution

Field Range	Count	Percentage	Real-world Analogy
1–10 fields	6	8.0%	Simple config files
11–25 fields	22	29.3%	API responses
26–50 fields	**27**	**36.0%**	**Most common**
51–100 fields	16	21.3%	Complex documents
100+ fields	4	5.3%	Large schemas

A.2 Field Type and Complexity Analysis

Tables 3 and 4 provide detailed breakdowns of field types and complexity metrics across the dataset.

Table 3: Distribution of Field Types Across All Samples

Field Type	Count	Percentage	Coverage
String	7,276	68.2%	Text data, IDs, descriptions
Integer	1,968	18.4%	Counts, IDs, numeric values
Array	907	8.5%	Lists, collections
Object	524	4.9%	Nested structures

Table 4: Statistical Summary of Complexity Metrics

Metric	Min	Max	Mean	Std Dev	Interpretation
Max Depth	2	7	4.0	1.0	Good variety
Total Fields	4	228	42.3	37.8	Wide range
Total Nodes	10	320	64.2	44.9	Diverse complexity
Arrays	0	11	4.7	2.7	Adequate coverage
Nested Objects	0	98	12.1	13.6	Strong variety

B Statistical Analysis Details

B.1 Methodology

We performed comprehensive statistical validation using:

- **Paired t-tests**: Each of the 75 base samples was evaluated by all metrics on the same variations, enabling paired comparisons
- **Bonferroni correction**: Applied for multiple comparisons ($\alpha = 0.05/30 = 0.0017$)
- **Effect size calculation**: Cohen's d to quantify practical significance
- **Confidence intervals**: 95% CIs computed using bootstrap with 1000 iterations

B.2 Semantic Variations - Detailed Results

Table 5: Summary Statistics for Semantic Variations (N=750)

Metric	Mean	Std Dev	95% CI	Min	Max	Median
STED	0.9539	0.0393	[0.9511, 0.9567]	0.832	1.000	0.961
BERTScore	0.9582	0.0249	[0.9564, 0.9600]	0.871	1.000	0.963
TED	1.0000	0.0000	[1.0000, 1.0000]	1.000	1.000	1.000
DeepDiff	0.7991	0.1647	[0.7873, 0.8109]	0.412	1.000	0.825

Table 6: Pairwise Statistical Comparisons - Semantic Variations

STED vs	p-values by Variation Ratio										Mean p
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	
TED	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001***
BERTScore	.391	.197	.724	.721	.857	.973	.655	.575	.454	.454	.600 (ns)
DeepDiff	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001***

*** Significant after Bonferroni correction ($p < 0.0017$), ns = not significant

Interpretation: STED and BERTScore show statistically equivalent performance on semantic variations (mean $p = 0.600$), both correctly maintaining high similarity for semantic equivalents. TED's complete insensitivity (constant 1.0) and DeepDiff's over-sensitivity (mean 0.799) are both significantly different from STED ($p < 0.001$).

B.3 Expression Variations - Detailed Results

Table 7: Summary Statistics for Expression Variations (N=750)

Metric	Mean	Std Dev	95% CI	Min	Max	Median
STED	0.9812	0.0174	[0.9799, 0.9824]	0.918	1.000	0.985
BERTScore	0.9595	0.0267	[0.9576, 0.9614]	0.882	1.000	0.964
TED	1.0000	0.0000	[1.0000, 1.0000]	1.000	1.000	1.000
DeepDiff	0.8051	0.1644	[0.7934, 0.8169]	0.423	1.000	0.831

Table 8: Pairwise Statistical Comparisons - Expression Variations

STED vs	p-values by Variation Ratio										Mean p
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	
TED	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001***
BERTScore	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001***
DeepDiff	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001***

*** Significant after Bonferroni correction ($p < 0.0017$)

Interpretation: STED significantly outperforms all baselines on expression variations ($p < 0.001$), demonstrating superior format-agnostic understanding. The lower standard deviation (0.0174) compared to semantic variations (0.0393) indicates more consistent performance.

B.4 Effect Size Analysis

Table 9: Cohen's d Effect Sizes for STED Comparisons

Comparison	Semantic Variations		Expression Variations	
	Cohen's d	Interpretation	Cohen's d	Interpretation
STED vs TED	-1.23	Large	-1.45	Large
STED vs BERTScore	-0.12	Negligible	0.82	Large
STED vs DeepDiff	1.26	Large	1.38	Large

Cohen's d interpretation: 0.2 = small, 0.5 = medium, 0.8 = large

B.5 Statistical Power Analysis

With 75 samples per variation ratio and 10 ratios per variation type:

- **Statistical power**: 0.99 for detecting medium effect sizes (d = 0.5) at $\alpha = 0.05$
- **Minimum detectable effect**: d = 0.33 with 80% power
- **Sample size adequacy**: Our 750 samples per variation type exceed requirements for robust conclusions

B.6 Schema Variation Robustness

We evaluate metric robustness under common schema evolution scenarios that preserve semantic equivalence but alter structural representation.

B.6.1 Field Name Evolution

Real-world schemas evolve through refactoring, where field names change while preserving meaning (e.g., API versioning). We simulate this by progressively renaming fields to semantically equivalent alternatives.

Table 10: Metric robustness to schema field renaming (N=75)

Fields Renamed	TED	STED	BERTScore	DeepDiff
10%	0.942	0.903	0.996	0.947
20%	0.915	0.893	0.995	0.942
30%	0.900	0.886	0.994	0.938
40%	0.879	0.882	0.993	0.933
50%	0.848	0.877	0.992	0.928
60%	0.831	0.874	0.991	0.923
70%	0.811	0.870	0.990	0.919
80%	0.790	0.866	0.989	0.916
90%	0.775	0.862	0.989	0.915
100%	0.759	0.856	0.988	0.913
Degradation[a]	-19.4%	-5.2%	-0.8%	-3.6%

[a] Relative decrease from 0% renamed (baseline=1.0) to 100% renamed

B.6.2 Schema Restructuring Patterns

We test two common schema refactoring patterns that preserve information content:

Table 11: Metric sensitivity to schema restructuring patterns

Metric	Schema Flattening[a]	Schema Nesting[b]
TED	0.000	0.506
STED	0.051	0.000
BERTScore	0.897	0.989
DeepDiff	0.809	0.781

[a] **Flattening**: Denormalizing nested objects into flat structure
Example: `{"user": {"name": "John", "age": 30}}` → `{"user_name": "John", "user_age": 30}`

[b] **Nesting**: Organizing flat fields into logical groups
Example: `{"street": "Main", "city": "NYC"}` → `{"address": {"street": "Main", "city": "NYC"}}`

The results demonstrate that:

- **TED** shows steep degradation (-19.4%) as field names change, treating renamed fields as completely different
- **STED** maintains better robustness (-5.2% degradation) through structural pattern recognition
- **BERTScore** demonstrates exceptional robustness (-0.8%) via semantic understanding of field names
- **DeepDiff** shows moderate robustness (-3.6%) with consistent degradation pattern

For structural transformations, TED and STED show complementary sensitivities: TED detects flattening as complete change (0.000) while STED identifies nesting as significant (0.000). This reflects their different approaches to structural similarity.

C Detailed Consistency Results

C.1 Temperature Sensitivity Analysis

We analyze how temperature settings affect model consistency across 9 temperature values from 0.1 to 0.9. Each cell represents mean consistency score across 80 test cases (720 total per model).

Table 12: Overall consistency scores across temperature settings

Model	Temperature								
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Claude-3.5-Sonnet-V2	0.595	0.666	0.628	0.607	0.603	0.604	0.577	0.578	0.570
Claude-3-Haiku	0.527	0.479	0.490	0.433	0.432	0.344	0.371	0.374	0.361
Claude-3.5-Haiku	0.674	0.650	0.537	0.530	0.513	0.497	0.448	0.442	0.428
Claude-3.7-Sonnet	**0.717**	**0.698**	**0.690**	**0.672**	**0.661**	**0.648**	**0.623**	**0.628**	**0.581**
Llama-3.3-70B	0.519	0.522	0.457	0.411	0.346	0.386	0.368	0.374	0.334
Nova-Pro	0.438	0.507	0.454	0.457	0.494	0.448	0.380	0.420	0.401

Table 13: Semantic consistency scores across temperature settings

Model	Temperature								
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Claude-3.5-Sonnet-V2	0.464	0.535	0.481	0.464	0.463	0.450	0.423	0.432	0.416
Claude-3-Haiku	0.456	0.406	0.402	0.337	0.336	0.263	0.283	0.279	0.264
Claude-3.5-Haiku	**0.583**	0.537	0.443	0.424	0.427	0.380	0.354	0.339	0.319
Claude-3.7-Sonnet	0.565	**0.547**	**0.535**	**0.508**	**0.498**	**0.476**	**0.454**	**0.462**	**0.414**
Llama-3.3-70B	0.387	0.390	0.326	0.292	0.222	0.263	0.246	0.257	0.215
Nova-Pro	0.342	0.404	0.354	0.348	0.377	0.338	0.283	0.314	0.300

Table 14: Structural consistency scores across temperature settings

Model	Temperature								
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Claude-3.5-Sonnet-V2	0.875	0.975	0.963	0.950	0.950	0.963	0.950	0.938	0.950
Claude-3-Haiku	0.755	0.750	0.781	0.767	0.758	0.679	0.682	0.708	0.722
Claude-3.5-Haiku	0.860	0.869	0.776	0.768	0.757	0.771	0.703	0.722	0.704
Claude-3.7-Sonnet	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**0.988**
Llama-3.3-70B	0.825	0.863	0.825	0.800	0.800	0.850	0.825	0.838	0.838
Nova-Pro	0.722	0.775	0.700	0.739	0.787	0.749	0.658	0.748	0.673

C.2 Summary Statistics

Table 15: Aggregated consistency metrics (Mean ± SD across all temperatures, N=720 per model)

Model	Overall	Semantic	Structural
Claude-3.5-Sonnet-V2	0.603±0.303	0.459±0.318	0.946±0.227
Claude-3-Haiku	0.424±0.386	0.336±0.399	0.734±0.432
Claude-3.5-Haiku	0.524±0.368	0.423±0.378	0.770±0.411
Claude-3.7-Sonnet	**0.658±0.240**	**0.495±0.288**	**0.999±0.037**
Llama-3.3-70B	0.413±0.337	0.289±0.340	0.829±0.377
Nova-Pro	0.444±0.378	0.340±0.388	0.728±0.430

C.3 Key Findings

Temperature Effects

- Most models show declining consistency as temperature increases, with steepest drops between T=0.5 and T=0.7
- Claude-3.5-Haiku exhibits strongest temperature sensitivity (46% decline from T=0.1 to T=0.9)
- Claude-3.7-Sonnet maintains most stable performance across temperatures (19% decline)
- Structural consistency remains more robust to temperature changes than semantic consistency

Optimal Temperature Ranges

- **T=0.1-0.3**: Best for consistency, all models achieve peak performance
- **T=0.4-0.6**: Moderate degradation, acceptable for most applications
- **T=0.7-0.9**: Significant consistency loss, especially for semantic preservation

Model-Specific Observations

- **Claude-3.7-Sonnet**: Near-perfect structural consistency ($\geq$0.988) across all temperatures
- **Nova-Pro**: Shows irregular pattern with local maximum at T=0.5
- **Llama-3.3-70B**: Exhibits sharp semantic consistency drop at T=0.5