A Reproducible Semantic Benchmark for Multivendor DSM-to-CLI Translation

Jerônimo Menezes[1,2], **Leonardo Bitzki**[1,2], **Diego Kreutz**[1]
Gefte Almeida[1], **Marcio Pohlmann**[1,2], **Rodrigo Mansilha**[1]

[1]AI Labs | PPGES, Universidade Federal do Pampa (UNIPAMPA)

[2]CPD, Universidade Federal do Rio Grande do Sul (UFRGS)

{jeronimomenezes,leonardobitzki,geftealmeida,marciopohlmann}.aluno@unipampa.edu.br

{diegokreutz,rodrigomansilha}@unipampa.edu.br

Abstract. *Translating high-level network intents into correct multivendor configurations remains challenging, as syntactically valid outputs may diverge from intended behavior. This paper presents a DSM-to-CLI semantic benchmark with multiple LLM translators, vendors, use cases, and repeated runs, using fixed judges and a failure taxonomy. Results show that semantic quality and reliability should be evaluated separately, vendor effects dominate use-case variation, and outcome dispersion correlates with vote instability. Huawei VRP scenarios expose vendor-specific failures not captured by aggregate metrics. Overall, multivendor benchmarking supports comparison of LLM-based configuration systems, while highlighting the need for complementary validation.*

1. Introduction

Network configuration in heterogeneous, multivendor environments remains a persistent operational challenge in modern network management. Differences in command syntax, vendor-specific semantics, feature support, and configuration abstractions make automation pipelines difficult to generalize and susceptible to silent failures [Boateng et al. 2025, Hong et al. 2025, Wang et al. 2026]. In practice, syntactically valid configurations do not necessarily implement the intended operational behavior, particularly when equivalent policies must be mapped across distinct vendor ecosystems. This gap between syntactic validity and semantic correctness remains a central obstacle to robust network automation [Wei et al. 2025, Tageldien et al. 2025].

Recent advances in Large Language Models (LLMs) have renewed interest in intent-driven network automation, with multiple studies exploring the translation of natural language intents or structured specifications into device-specific configurations [Menezes et al. 2025, Wang et al. 2024, Mendoza and Ocampo 2025, Menezes et al. 2026]. These results demonstrate the practical feasibility of LLM-based translation pipelines. However, as highlighted by recent surveys [Boateng et al. 2025, Hong et al. 2025], the current state of the art remains methodologically fragmented. Many existing approaches still emphasize prompt engineering, lexical similarity, single-pass execution success, or narrow functional checks, while providing limited evidence about semantic fidelity, reproducibility, stability across executions, and comparability across models and vendors.

This limitation is particularly relevant in Intent-Based Networking (IBN), where the objective is not merely to generate plausible commands, but to produce vendor-specific configurations that preserve the intended network state [Wei et al. 2025]. In such settings, evaluation based only on syntactic plausibility, string matching, or isolated functional probes may overlook semantic mismatches. Multiple command sequences may be syntactically different yet semantically equivalent, while seemingly correct outputs may still violate the declared intent. Consequently, progress in this domain benefits from semantically grounded benchmarks, independent semantic assessment, repeated-run analysis, and controlled multivendor experimentation [Tageldien et al. 2025, Raptis et al. 2025].

In previous work, we introduced *dsm2cli*[1], an observable pipeline for translating Desired State Models (DSMs) into multivendor CLI. The framework separates translation from semantic assessment, uses independent LLM evaluators, and records structured artifacts such as verdicts and failure evidence. That study showed that semantic failures may remain hidden in generation-only workflows and that explicit assessment improves interpretability, without constituting formal or execution-based validation.

Building on this, we introduce *dsm2cli-bench*[2], a reproducible framework for evaluating DSM-to-CLI translation. It defines a controlled semantic benchmark across multiple vendors, use cases, fixed judges, repeated runs, and a failure taxonomy. Rather than focusing only on aggregate accuracy, the benchmark enables multidimensional evaluation, including semantic correctness, stability, agreement, vendor sensitivity, latency, and token efficiency.

The main contributions of this paper are fourfold: first, we design a reproducible semantic benchmark for DSM-to-CLI translation in multivendor environments; second, we introduce a controlled experimental protocol that combines structured DSM inputs, fixed LLM-based judges, repeated executions, and an explicit failure taxonomy; third, we provide a comparative evaluation of five cloud LLM translators across Cisco NX-OS, Huawei VRP, and Arista EOS; and finally, we present a multidimensional analysis encompassing semantic correctness, execution stability, inter-evaluator agreement, latency, and token usage.

2. Related Work

Recent studies and surveys have shown growing interest in the use of Large Language Models (LLMs) for network management and configuration tasks, particularly through pipelines that translate natural language intents, structured specifications, or intermediate representations into vendor-specific commands and executable workflows [Long et al. 2025, Boateng et al. 2025, Hong et al. 2025, Liu et al. 2024]. However, existing work remains heterogeneous across input representations, vendor coverage, evaluation methodology, repeated-run analysis, and benchmark reproducibility, as summarized in Table 1.

The analyzed literature shows that most existing approaches focus on translation pipelines, typically NL→CLI, NL→IR→CLI, or structured specification→configuration

[1] `https://github.com/net2d-community/dsm2cli`
[2] `https://github.com/net2d-community/dsm2cli-bench`

workflows, as in Net2d-LLM [Menezes et al. 2025], NetConfEval [Wang et al. 2024], PeeringLLM-Bench [Mendoza and Ocampo 2025], and INTA [Wei et al. 2025]. These studies demonstrate the feasibility of LLM-assisted network configuration, but they differ substantially in input structure, validation strategy, execution environment, and artifact preservation. As a result, direct comparison across models, vendors, and configuration domains remains difficult.

A second relevant limitation concerns vendor coverage. Many prior studies operate in single-vendor or narrowly scoped environments, where heterogeneity is only partially represented. Even works explicitly targeting multivendor settings often focus on a specific protocol, configuration family, or evaluation context, such as BGP peering, configuration translation, or controlled execution tasks [Mendoza and Ocampo 2025, Wei et al. 2025]. Recent surveys further identify broad multivendor validation as an open challenge for LLM-based network automation [Hong et al. 2025, Boateng et al. 2025].

Table 1. Comparative analysis of LLM-based network configuration approaches.

Paper	Input → Output	Multivendor	Semantic Evaluation	Repeated Runs	Benchmark Protocol
Net2d-LLM	DSM → CLI	No	No	No	Partial
NetConfEval	NL → Spec/API/CLI	No	Lexical/syntactic	Limited	Partial
PeeringLLM-Bench	NL → CLI	Yes	Binary/functional	Yes	Yes
NetLLMBench	NL → Execution	No	No	No	Yes
INTA	CLI → CLI	Limited	No	No	No
dsm2cli-bench (ours)	**DSM → CLI**	**Yes**	**Independent LLM-based**	**Yes**	**Yes (reproducible)**

Semantic evaluation remains a key gap. Existing studies rely on textual similarity, syntactic validation, execution-based proxies, or human inspection [Wang et al. 2024, Aykurt et al. 2024], but these do not always capture whether configurations preserve the intended network state. This is critical in multivendor settings, where different commands may express equivalent semantics, while plausible outputs may omit required behavior.

Another challenge is the limited analysis of repeated runs and stability. As LLM outputs vary across executions, single-run experiments provide weak evidence on reproducibility and consistency. Although some benchmarks include repetition, stability is not consistently treated as a primary dimension [Boateng et al. 2025, Hong et al. 2025, Long et al. 2025].

In contrast, our work proposes a reproducible DSM→CLI benchmark combining structured inputs, fixed LLM judges, repeated runs, failure labeling, and multivendor comparison. Rather than ensuring formal correctness, it provides a controlled semantic assessment protocol and preserves artifacts for analyzing translator behavior, disagreement, errors, latency, and token usage.

3. Benchmark Design and Methodology

This section presents the benchmark design, experimental protocol, evaluation metrics, and statistical procedures adopted to assess DSM-to-CLI semantic translation in multivendor environments.

3.1. Research Questions and Hypotheses

The benchmark investigates three dimensions of structured-intent translation into vendor-specific CLI: vendor-dependent behavior, use-case-dependent difficulty, and stability un-

der repeated executions. To make these dimensions operational, we define three hypotheses and their corresponding planned evidence in Table 2.

Table 2. Operational hypotheses and planned evidence.

Hyp.	Operational formulation	Planned evidence
H1	Huawei VRP exhibits lower end-to-end correctness than Cisco NX-OS and Arista EOS, while Cisco and Arista show similar behavior.	Planned vendor contrasts using correctness proportions, Wilson confidence intervals, and two-proportion z-tests.
H2	SVI-based scenarios exhibit lower correctness and/or lower evaluator agreement than Layer 2 scenarios.	Planned Layer 2 versus SVI contrast, complemented by agreement analysis stratified by use case.
H3	Experimental cells with higher repeated-run outcome dispersion exhibit lower vote stability.	Cell-level regression of the Vote Stability Index (VSI) on the standard deviation of binary outcomes across repetitions.

These hypotheses are evaluated as benchmark-specific claims. They are not intended to establish general properties of vendors, models, or network operating systems beyond the experimental scope considered in this study.

3.2. Experimental Harness

The benchmark adopts *dsm2cli* as a controlled experimental harness. For each execution, the system receives a DSM, device metadata, and a selected translator model, producing vendor-specific CLI that is subsequently assessed by a fixed panel of three independent LLM-based judges. Their individual votes are aggregated into a final semantic verdict, while all intermediate artifacts are persistently recorded.

This design separates CLI generation from semantic assessment, ensuring a stable input–output contract across translators. It also preserves structured artifacts, including generated CLI, individual judge votes, final verdicts, failure labels, and execution telemetry. Throughout all experiments, the DSM schema, judge panel, failure taxonomy, and telemetry pipeline remain fixed, with the translator model as the primary experimental factor. The assessment layer should be interpreted as an independent semantic evaluation protocol, not as formal verification or execution-based validation.

3.3. Experimental Matrix

The experimental matrix spans three vendors, Cisco NX-OS, Huawei VRP, and Arista EOS, across five representative use cases: UC1 (L2 access), UC2 (L2 trunk with native VLAN), UC3 (SVI IPv4), UC4 (SVI IPv6), and UC5 (dual-stack SVI).

These use cases were selected as a controlled initial benchmark because they cover common Layer 2 and Layer 3 interface configuration primitives while remaining compact enough for repeated, cross-vendor semantic assessment. They include both access/trunk switching behavior and routed virtual interface behavior, allowing the benchmark to compare relatively simple L2 tasks with SVI-based tasks involving IPv4, IPv6, and dual-stack addressing. More complex domains, such as ACLs, QoS, routing policies, and VPN services, are intentionally left outside the present scope and treated as future extensions.

The DSM is treated as the complete normative specification. No prior device state is assumed, and vendor defaults are considered valid only when explicitly encoded in the

DSM. This yields 15 scenarios per translator (3 vendors $\times$ 5 use cases). Each scenario is executed 10 times, resulting in 150 runs per translator and 750 executions overall.

3.4. Translators and Judges

Five cloud-based translators are evaluated: `gpt-5`, `claude-opus-4-6`, `gemini-2.5-pro`, `deepseek-chat`, and `grok-4-1-fast-reasoning`, covering distinct provider families under a common semantic translation protocol.

Semantic assessment is performed by a fixed panel of three judges: `gpt-5.4-mini`, `claude-haiku-4-5`, and `gemini-2.5-flash-lite`. This configuration avoids self-evaluation and reduces evaluator variation across translators, vendors, and repeated executions. The same judge panel is used throughout the experiment to preserve comparability across all experimental cells.

3.5. Execution Protocol

Each execution is defined by the tuple (translator, vendor, use case, repetition). Experiments are organized into 10 rounds, in which all 75 experimental cells (5 translators $\times$ 3 vendors $\times$ 5 use cases) are executed once per round in randomized order. This procedure reduces the risk that temporal effects or provider-side instability systematically favor a specific translator, vendor, or use case.

Translator temperature is fixed at 0.2, while judge temperature is set to 0.0 to favor deterministic assessment. Maximum output lengths are capped at 768 tokens for translators and 1536 tokens for judges. A timeout threshold of 300 seconds is enforced, with up to two automatic retries for technical failures. Runs that remain unsuccessful after retries are classified as pipeline errors.

3.6. Metrics and Failure Taxonomy

The benchmark collects three classes of metrics: semantic quality, consistency, and performance. Semantic quality is quantified through correctness rate, mean favorable votes, semantic failure rate, and pipeline error rate. Consistency is measured at the cell level using vote dispersion and the Vote Stability Index (VSI), defined as the proportion of repetitions assigned to the modal vote category. Performance is assessed through mean latency, 95th-percentile latency, and token usage.

Semantic failures are classified as `omission`, `divergence`, and `structural`. Pipeline failures are categorized as `timeout`, `provider_error`, `invalid_output`, `parsing_failure`, and `empty_response`. This taxonomy separates failures in the generated configuration from technical failures in the translation or evaluation pipeline.

3.7. Statistical Analysis

The statistical analysis is specified *a priori*. Hypotheses H1 and H2 are evaluated through planned proportion contrasts using Wilson confidence intervals and two-proportion z-tests. H1 compares correctness across vendors, while H2 compares Layer 2 and SVI-based scenarios.

Hypothesis H3 is assessed at the cell level through ordinary least squares regression:

$$\text{VSI} \sim \texttt{sd\_outcome}$$

where `sd_outcome` denotes the standard deviation of the binary correct/not-correct outcome across the 10 repetitions of each experimental cell. Mean accuracy is intentionally excluded from this model to avoid structural coupling between correctness and stability. This analysis is interpreted as descriptive evidence of association, not as a causal model. Inter-judge agreement is quantified using Fleiss' Kappa and percent agreement, stratified by vendor and use case.

3.8. Pilot Calibration

A pilot phase was conducted to validate the benchmark implementation and led to two protocol refinements: improved semantic equivalence handling for UC2 under vendor-specific CLI syntax and explicit `pipeline_error_type` labeling to improve observability of technical failures.

These refinements did not change the core experimental matrix, translator set, judge panel, or repeated-run protocol. They improved the consistency of the assessment procedure before the final experiment. In the final dataset, artifact inspection also identified 17 HTTP 400 errors from `claude-haiku-4-5` that had initially been logged as structural translator failures because the pipeline stored the judge error message as a semantic finding. These cases were subsequently reclassified as judge-side faults, separating evaluation-layer failures from genuine CLI-generation failures.

4. Experimental Results

This section presents benchmark results aggregated over ten rounds (750 executions), distinguishing two notions of correctness. End-to-end correctness considers all runs, including pipeline failures, while semantic correctness considers only valid judged runs, excluding pipeline and judge faults.

Analyses use different subsets of these executions (Table 3). End-to-end metrics are computed over all runs, while semantic metrics use the 667 valid judged runs. Inter-judge agreement is reported under full and binary-only views (Section 4.7). All raw data, including CLI outputs, judge verdicts, telemetry, and failure labels, are publicly available in the project repository, supporting reproducibility and inspection.

Table 3. Execution outcome summary and analysis subsets.

Outcome	Raw	Final	Rate
Correct (semantic)	655	655	87.3%
Semantic incorrect	29	12	1.6%
Judge fault (infra)	0	17	2.3%
Pipeline error	66	66	8.8%
Total	750	750	100.0%

4.1. Infrastructure Events and Outcome Classification

Before semantic analysis, two infrastructure-level events must be accounted for. First, starting in round 7, `claude-opus-4-6` experienced a sustained provider-side outage, resulting in 50 `provider_error` outcomes out of 150 scheduled runs. All 100 executions completed before the outage were semantically correct.

Second, 17 runs were affected by HTTP 400 credential errors in the judge model `claude-haiku-4-5`. These cases were initially logged as structural translator failures because the pipeline stored the judge error message as a semantic finding. Artifact inspection indicated that the fault originated in the evaluation layer rather than in the generated CLI, and these runs were therefore reclassified as *judge faults*. This reclassification separates evaluation-layer faults from genuine CLI-generation failures.

4.2. Overall Results

After reclassifying judge faults, the benchmark results comprise 655 correct executions, 12 genuine semantic failures, 17 judge faults, and 66 pipeline errors across 750 runs. Excluding infrastructure-related events, the semantic correctness rate over valid judged runs reaches 98.2%.

The 12 genuine semantic failures consist of 11 omissions and 1 divergence, with no translator-generated structural errors. This indicates that, within this benchmark, semantic failures arise mainly when the generated CLI omits required DSM elements rather than explicitly contradicting the declared intent.

4.3. Results by Translator

At the translator level, `grok-4-1-fast-reasoning` achieves the highest end-to-end correctness, with 149 correct executions out of 150 and no infrastructure-related disruptions (Table 4). In contrast, `claude-opus-4-6` achieves semantic correctness on all valid judged runs, but its end-to-end rate is substantially reduced by the provider-side outage that affected 50 executions. `deepseek-chat` performs strongly on Cisco and Arista, but its overall score is reduced by semantic failures and pipeline errors concentrated in Huawei scenarios.

Table 4. Translator-level correctness, semantic failures, and pipeline errors.

Translator	Corr./Tot.	Rate	Sem.	Pipe.	μ votes	σ votes
`grok-4-1-fast-reasoning`	149/150	99.3%	1	0	2.660	0.489
`gpt-5`	143/150	95.3%	3	0	2.507	0.588
`gemini-2.5-pro`	142/150	94.7%	3	0	2.520	0.599
`deepseek-chat`	121/150	80.7%	5	16	2.455	0.667
`claude-opus-4-6`	100/150	66.7% / 100%*	0	50	2.990	0.100

* Rate over valid non-outage runs.

These results show why semantic quality and operational reliability should be reported separately. In particular, `claude-opus-4-6` remains semantically correct on all valid non-outage runs despite ranking lower in the end-to-end view, as illustrated in Figure 1.

4.4. Results by Vendor

Vendor-level variation is the largest observed effect in the benchmark. As shown in Table 5, Cisco NX-OS and Arista EOS exhibit similar end-to-end behavior, whereas Huawei VRP concentrates most genuine semantic failures (11 of 12) and a larger share of pipeline errors (33 of 66). The planned contrasts in Table 6 provide support for H1 in this benchmark: the Cisco–Huawei and Arista–Huawei gaps are statistically significant, while Cisco and Arista are not significantly different.

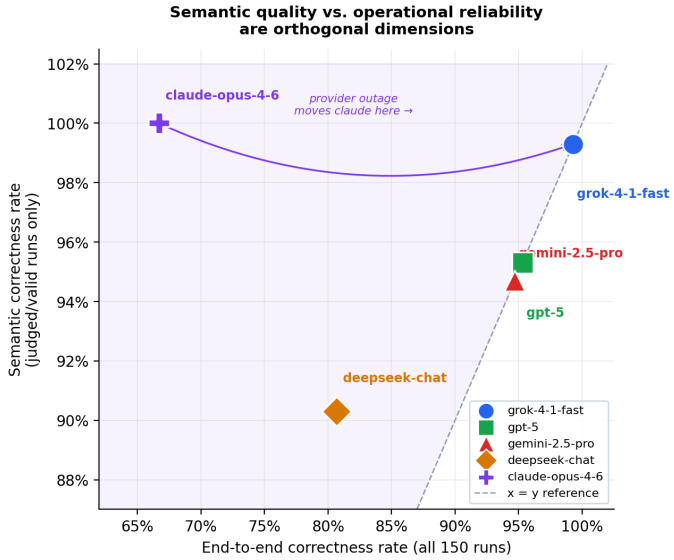

Figure 1. Semantic versus end-to-end correctness by translator.

The vendor effect amplitude reaches 17.6 p.p., which is 3.3× larger than the use-case effect amplitude (5.3 p.p.). This indicates that, under the present experimental design, vendor identity is a larger source of variation than the use-case grouping.

The most pronounced interaction is observed for `deepseek-chat` on Huawei VRP (Figure 2). While this translator achieves 100% correctness on Cisco and Arista, its performance drops sharply on Huawei, combining semantic failures in UC3 with invalid-output pipeline failures in UC4 and UC5. This pattern is consistent across rounds and could be obscured by aggregate cross-vendor summaries.

Table 5. Aggregate results by vendor.

Vendor	Corr./Tot.	Rate	SI	PE	μ votes
Cisco NX-OS	234/250	93.6%	0	16	2.705
Arista EOS	231/250	92.4%	1	17	2.674
Huawei VRP	190/250	76.0%	11	33	2.419

Table 6. Planned vendor contrasts for H1.

Contrast	$\hat{p}_1$	$\hat{p}_2$	Δ (95% CI) / p
Cisco vs Huawei	0.936	0.760	+0.176 [0.115, 0.237] / <0.001
Arista vs Huawei	0.924	0.760	+0.164 [0.102, 0.226] / <0.001
Cisco vs Arista	0.936	0.924	+0.012 [−0.033, 0.057] / 0.599

4.5. Results by Use Case

At the use-case level, the observed effect is weaker than the vendor dimension (Table 7). When all runs are considered, the contrast between Layer 2 and SVI scenarios does not reach statistical significance. After excluding the `claude-opus-4-6` outage, the gap

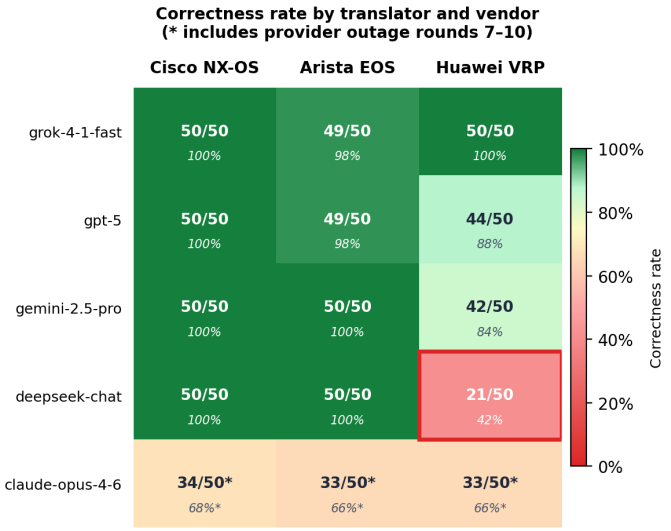

Figure 2. Correctness rate by translator and vendor.

becomes nominally significant (Table 8). Overall, these results provide directional but modest support for H2 and indicate that the use-case effect is sensitive to infrastructure-related disruptions.

Among the SVI scenarios, UC3 concentrates the largest number of semantic failures, largely driven by the `deepseek-chat` × Huawei interaction discussed earlier. UC4 exhibits more pipeline errors, but only one semantic failure, suggesting that its lower end-to-end rate is primarily operational rather than semantic. After pilot calibration, UC2 became the strongest use case in both end-to-end correctness and inter-judge agreement, suggesting that the calibration improved the consistency of the assessment protocol without altering the core benchmark design.

Table 7. Aggregate results by use case.

UC	Description	Corr./Tot.	Rate	SI	PE	μ votes
UC1	L2 access	132/150	88.0%	1	11	2.583
UC2	L2 trunk + VLAN	135/150	90.0%	3	10	2.629
UC3	SVI IPv4	130/150	86.7%	6	10	2.564
UC4	SVI IPv6	127/150	84.7%	1	20	2.638
UC5	Dual-stack SVI	131/150	87.3%	1	15	2.607

Table 8. Planned contrast for H2 (Layer 2 vs SVI).

Contrast	$\hat{p}_{L2}$	$\hat{p}_{SVI}$	Δ (95% CI)	p-value
L2 vs SVI (all runs)	0.890	0.862	+0.028 [−0.020, 0.075]	0.253
L2 vs SVI (excl. outage)	0.950	0.908	+0.042 [+0.001, 0.082]	0.044

4.6. Stability and Reliability

With ten repetitions per cell, the benchmark enables an assessment of execution stability. As a complementary analysis, we regress the Vote Stability Index (VSI) on the standard

deviation of the binary correct/not-correct outcome across the 75 experimental cells. The fitted model is:

$$\widehat{\text{VSI}} = 1.006 - 0.652 \cdot \text{sd\_outcome}, \quad R^2 = 0.922,\ t(73) = -29.4,\ p < 0.001$$

with a 95% confidence interval for the slope of $[-0.696,\ -0.609]$.

This association provides strong descriptive evidence consistent with H3: cells with greater outcome dispersion tend to exhibit less stable voting patterns (Figure 3). This relationship should not be interpreted causally, since both VSI and `sd_outcome` are derived from the same repeated outcome sequence and are therefore partially coupled by construction.

A relevant qualitative finding is that no cell simultaneously achieves perfect correctness and perfect vote stability. Even fully correct cells occasionally oscillate between 2/3 and 3/3 favorable votes, suggesting that part of the residual dispersion reflects evaluator uncertainty rather than translator instability alone.

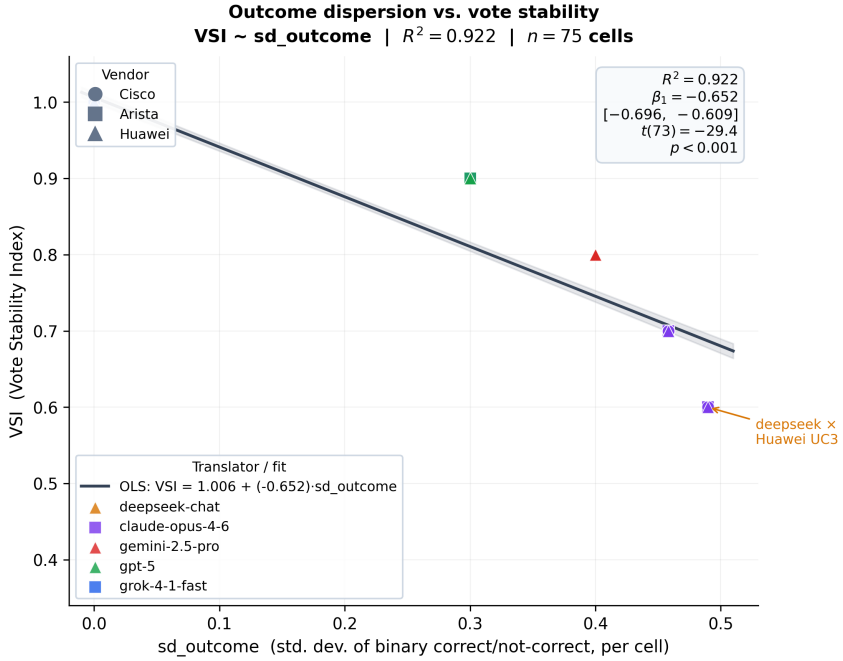

Figure 3. Outcome dispersion versus vote stability across 75 experimental cells.

4.7. Agreement Among Judges

Table 9 reports inter-judge agreement under two complementary views. **View A** includes all 667 runs that reached the judging stage, excluding only pipeline errors and reclassified judge faults. **View B** restricts the analysis to the 483 runs in which judge outputs can be reduced to a binary correct/incorrect decision. This second view excludes structurally mixed cases, such as executions in which one judge reported a `structural` finding while the other judges voted correct.

The two views provide different but complementary perspectives. In View A, all kappa values are negative, a consequence of the prevalence paradox under an imbalanced

class distribution: approximately 66% of runs are unanimous, increasing expected chance agreement and making κ difficult to interpret. These values should therefore not be read as direct evidence of poor judge consistency.

Table 9. Inter-judge agreement under full and binary-only views.

	View A (n=667)			**View B** (n=483)		
Stratum	n	κ	Splits	n	κ	Splits
Overall	667	−0.077	225	483	+0.197	41
Cisco NX-OS	234	−0.109	69	165	+1.000	0
Arista EOS	232	−0.104	73	162	+0.244	3
Huawei VRP	201	−0.046	83	156	+0.134	38
UC1	133	−0.106	45	98	+0.056	10
UC2	138	−0.060	45	98	+0.358	5
UC3	136	−0.019	47	100	+0.314	11
UC4	128	−0.100	42	91	+0.148	5
UC5	132	−0.111	46	96	+0.055	10

In View B, after restricting the analysis to binary decisions, kappas become positive and more informative. The vendor-level pattern remains consistent across both views: Huawei VRP concentrates the largest number of splits (83 in View A and 38 in View B), suggesting that this vendor is more difficult to evaluate consistently under the adopted judging protocol. The perfect kappa observed for Cisco NX-OS in View B should also be interpreted cautiously, since mixed structural cases are excluded from that view.

4.8. Latency and Token Usage

Operational telemetry indicates no direct correlation between correctness and latency in this benchmark. `grok-4-1-fast-reasoning` combines the highest end-to-end correctness (99.3%) with moderate latency (μ=31.9 s; 7,834 total tokens), making it the most efficient translator under the evaluated conditions. By contrast, `gpt-5` achieves similarly high correctness (95.3%) at 1.9× higher mean latency (μ=60.4 s, p95 = 120.8 s) and the highest token cost (9,956 tokens/run).

The higher latency of `gpt-5` is associated with longer CLI outputs (3,585 translator tokens versus 1,402–1,552 for the other models), which also increases downstream judge evaluation cost. These results expose a practical deployment trade-off: higher semantic reliability may come with greater latency and token cost, depending on the translator and provider behavior.

For most translators, the judge layer dominates total token consumption (81–82%). This indicates that LLM-based semantic assessment should be treated as an explicit operational cost in deployment scenarios, rather than as a negligible post-processing step.

5. Discussion

The results indicate that aggregate correctness alone is insufficient for evaluating LLM-based DSM-to-CLI systems; a multidimensional view is required, separating semantic

correctness, reliability, stability, judge agreement, latency, and token cost. Semantic quality and operational reliability behave independently: `claude-opus-4-6` achieved full semantic correctness on valid runs but reduced end-to-end correctness due to provider failures, while high availability may still mask semantic errors.

Vendor-specific effects are central. Cisco NX-OS and Arista EOS show similar behavior, whereas Huawei VRP concentrates most failures and disagreement, indicating that vendor identity strongly influences both translation and evaluation under a fixed protocol. Repeated executions further expose this sensitivity: the `deepseek-chat` $\times$ Huawei case reveals failures not observed on other vendors, which would be obscured by aggregate results, highlighting the importance of stability analysis.

Benchmark calibration also plays a critical role. The UC2 refinement improved consistency in handling VLAN semantics, showing that equivalence definitions directly affect outcomes and must be fixed prior to final evaluation. At the same time, LLM-based judges remain a limitation: despite fixed panels, artifact preservation, and agreement analysis, they do not guarantee correctness. The benchmark therefore measures semantic adherence under a controlled protocol, not formal or production-level validity, requiring complementary validation methods.

Finally, latency and token results show that semantic assessment is operationally significant, with the judge layer dominating cost. Practical deployments must balance assurance, cost, and response time through strategies such as lighter judges, escalation, and hybrid validation. Overall, evaluation should focus not only on average correctness, but on failure modes, vendor sensitivity, stability, agreement, and cost.

6. Threats to Validity

Internal validity is affected by runtime conditions outside experimental control, including provider outages, queueing, transport failures, and API errors, which impact latency, availability, and end-to-end correctness independently of translator capability. We mitigate this by separating `pipeline_error`, `judge_fault`, and `semantic_incorrect` outcomes and by randomizing execution order, although infrastructure events may still affect comparability across translators. Another internal threat concerns alignment between the intended semantic protocol and its implementation: pilot results revealed overly literal judge behavior for VLAN constructs, motivating refinement of UC2 equivalence handling. While this improved consistency, results remain sensitive to the definition of semantic equivalence; to reduce this risk, the final protocol fixes the DSM schema, judge panel, failure taxonomy, and repeated-run matrix.

Construct validity is limited by measuring semantic adherence under an LLM-based assessment protocol rather than operational correctness in live networks. The benchmark does not perform formal verification, device-state reconciliation, or execution-based validation, and evaluates semantic consistency with the DSM according to LLM judges. This introduces additional sources of uncertainty, as judges may be influenced by prompt sensitivity, bias, vendor-specific syntax interpretation, and provider-side changes. Experimental design choices such as translators, judges, temperature settings, token limits, timeouts, and retry policies may also affect the results. Although all parameters are fixed across runs and artifacts are preserved, no sensitivity analysis is conducted; therefore, the findings should be interpreted as valid for the evaluated protocol rather than as

general rankings of LLMs.

External validity is bounded by the benchmark scope, which includes three vendors and five use cases focused on Layer 2 and SVI-based configurations. Other domains (e.g., ACLs, QoS, routing, VPNs, multicast, and security policies) are not covered, and the assumption that the DSM fully specifies the target state simplifies real deployments, where configurations interact with existing device state and policies. Finally, reproducibility is affected by benchmark drift and provider-side model evolution: although all artifacts are preserved, cloud-hosted models may change over time, preventing exact re-execution and requiring longitudinal evaluation to assess stability of conclusions.

7. Conclusions and Future Work

This paper presented a reproducible semantic benchmark for comparing cloud LLM translators in DSM-to-CLI conversion. The benchmark spans five translators, three vendors, five use cases, and ten repetitions per cell (750 executions), under a controlled protocol with fixed judges, failure labeling, and artifact preservation.

Results support a multidimensional evaluation. Vendor effects dominated use-case variation, with Huawei VRP concentrating most failures and disagreement, supporting H1. H2 received limited support, as Layer 2 versus SVI differences were weak and sensitive to infrastructure. H3 showed strong support: higher outcome dispersion was associated with lower vote stability, although both derive from the same runs.

Three lessons emerge. First, semantic correctness and operational reliability should be reported separately, as infrastructure failures may obscure performance. Second, aggregate metrics may hide translator-by-vendor effects, as in the `deepseek-chat` $\times$ Huawei VRP case. Third, LLM-based assessment improves transparency and scalability, but does not replace formal or execution-based validation. The benchmark should be viewed as a controlled semantic assessment protocol rather than proof of correctness. Its main contribution is methodological, providing a reproducible framework to analyze correctness, failures, stability, agreement, latency, and token usage.

Future work will extend the benchmark to additional vendors and domains (e.g., ACLs, QoS, routing policies, VPNs), incorporate complementary validation (e.g., digital twins, rule-based methods, expert review), evaluate alternative translators and judge settings, and study benchmark drift and model evolution through longitudinal re-execution.

Acknowledgments

This work was partially supported by the Brazilian National Council for Scientific and Technological Development (CNPq) under Grant 409743/2025-9; by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), Finance Code 001; and by the Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul (FAPERGS) under Grants 24/2551-0001368-7, 24/2551-0000726-1, and 25/2551-0002572-9.

References

Aykurt, K., Blenk, A., and Kellerer, W. (2024). NetLLMBench: A benchmark framework for large language models in network configuration tasks. In *IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. IEEE.

Boateng, G. O., Sami, H., Alagha, A., Elmekki, H., Hammoud, A., Mizouni, R., Mourad, A., Otrok, H., Bentahar, J., Muhaidat, S., et al. (2025). A survey on large language models for communication, network, and service management: Application insights, challenges, and future directions. *IEEE Communications Surveys & Tutorials*.

Hong, J., Tu, N. V., and Hong, J. W.-K. (2025). A comprehensive survey on LLM-based network management and operations. *International Journal of Network Management*, 35(6):e70029.

Liu, C., Xie, X., Zhang, X., and Cui, Y. (2024). Large language models for networking: Workflow, advances, and challenges. *IEEE Network*, 39(5):165–172.

Long, S., Tan, J., Mao, B., Tang, F., Li, Y., Zhao, M., and Kato, N. (2025). A survey on intelligent network operations and performance optimization based on large language models. *IEEE Communications Surveys & Tutorials*, 27(6):3915–3949.

Mendoza, J. R. and Ocampo, R. (2025). PeeringLLM-Bench: Evaluating LLMs for BGP configuration tasks. In *Proceedings of the 20th Asian Internet Engineering Conference*.

Menezes, J., Bitzki, L., and Kreutz, D. (2025). Net2d-LLM: Translating Structured Network Intents into CLI using LLMs with Execution in a Network Digital Twin. In *Anais da XXII ERRC*. SBC.

Menezes, J., Bitzki, L., Kreutz, D., Almeida, G., Pohlmann, M., and Mansilha, R. (2026). dsm2cli: An observable pipeline for translating network intents into multivendor CLI with independent semantic assessment. In *Anais do Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*. SBC.

Raptis, N., Adhane, G., Fonseca, J. P., Ramantas, K., and Verikoukis, C. (2025). ARGVI: Adaptive routing, generation, and validation of intents for intent-driven management. In *2025 IEEE Conference on Network Function Virtualization and Software-Defined Networking (NFV-SDN)*, pages 1–6.

Tageldien, M., Selim, B., and Sboui, L. (2025). Large language models in intent-based networking: A comprehensive survey across the intent lifecycle. In *ITC-Egypt*, pages 810–817. IEEE.

Wang, C., Scazzariello, M., Farshin, A., Ferlin, S., Kostić, D., and Chiesa, M. (2024). NetConfEval: Can LLMs facilitate network configuration? *Proceedings of the ACM on Networking*, 2(CoNEXT2):1–25.

Wang, J., He, B., Zhao, J., Xuan, Y., Sun, H., Qi, Q., Liang, J., Zhuang, Z., and Liao, J. (2026). LLM-powered intent-driven configuration generation for multi-vendor networks. *IEEE Transactions on Network and Service Management*, PP:1–1.

Wei, Y., Xie, X., Hu, T., Zuo, Y., Chen, X., Chi, K., and Cui, Y. (2025). INTA: Intent-Based Translation for Network Configuration with LLM Agents. In *2025 IEEE 33rd International Conference on Network Protocols (ICNP)*, pages 1–16, Seoul, Korea, Republic of. IEEE.