Framework para Verificação e Validação Experimental de Configurações de Rede Geradas por LLMs

Cristiano da Silveira Colombo[1 2], **Magnos Martinello**[1]

[1]Universidade Federal do Espírito Santo (UFES)

magnos@inf.ufes.br

[2]Instituto Federal do Espírito Santo (IFES)

cristianos@ifes.edu.br

Abstract. *The use of Large Language Models (LLMs) for generating network configurations and validating policies has gained increasing attention in the literature. Unlike the predominant approach that prioritizes generation accuracy, this work directs its efforts toward the verification process, evaluating stability under load (batch size) and the complexity of intents translated from natural language into technical requirements. This paper presents three main contributions: the proposal of a multi-stage architecture that integrates logical inconsistency detection at the LLM output stage; the identification of model reliability limits relative to task batch size; and the demonstration that syntactic validity is an insufficient indicator for ensuring operational connectivity in computer networks. The results show that the volume of simultaneous processing is a determining factor in correctness degradation, suggesting a critical inflection point around 20 tasks per batch. It was also observed that the structural precision of the models is inversely proportional to the context window density. This phenomenon indicates that the LLM suffers from focus dilution, tending to omit essential logical parameters, which invalidates the configuration at the data plane.*

Resumo. *A utilização de Modelos de Linguagem de Grande Porte (LLMs) na geração de configurações e validação de políticas de rede tem recebido atenção crescente na literatura. Diferente da abordagem predominante que prioriza a acurácia da geração, este trabalho direciona seus esforços para o processo de verificação, avaliando a estabilidade sob carga (batch size) e a complexidade das intenções traduzidas de linguagem natural para requisitos técnicos. O artigo apresenta três contribuições : a proposição de uma arquitetura multi-estágio que integra a detecção de inconsistências lógicas à saída dos LLMs ; a identificação do limite de confiabilidade dos modelos em relação ao tamanho do lote de tarefas; e a demonstração de que a validade sintática é um indicador insuficiente para garantir a conectividade operacional em redes de computadores. Os resultados evidenciam que o volume de processamento simultâneo é determinante na degradação da corretude, sugerindo a existência de um ponto de inflexão crítico em torno de 20 tarefas por lote. Observou-se também que a precisão estrutural dos modelos é inversamente proporcional à densidade da janela de contexto. Este fenômeno indica que o LLM sofre uma diluição de foco, tendendo a omitir parâmetros lógicos essenciais, o que invalida a configuração no data plane.*

1. Introdução

A gerência de redes baseada em intenção (*Intent-Based Networking* – IBN) busca simplificar a administração de infraestruturas complexas ao permitir que requisitos de alto nível sejam traduzidos em configurações de dispositivos [Tu et al. 2024, Hossain and Aljoby 2025]. O uso de Modelos de Linguagem de Grande Porte (LLMs) demonstrou potencial para acelerar essa tradução, convertendo descrições em linguagem natural para formatos estruturados como JSON ou XML [Kou et al. 2025, Zheng et al. 2024].

A adoção dessas tecnologias em ambientes de produção enfrenta um desafio de confiabilidade operacional [Zhou et al. 2025]. Observa-se que a conformidade sintática do código gerado por um LLM não garante que a configuração seja funcional ou livre de erros lógicos [Bhardwaj and Fomina 2025]. Os LLMs podem produzir arquivos estruturalmente corretos, mas que falham ao lidar com interdependências de protocolos, como conflitos de endereçamento IP ou rotas tecnicamente impossíveis. Este cenário cria uma lacuna entre o artefato gerado e sua execução efetiva no data plane [Hachimi et al. 2025].

A literatura recente concentra seus esforços na avaliação de métricas de qualidade de geração e similaridade textual [Aykurt et al. 2024, Anwar and Caesar 2024], deixando em segundo plano a garantia de que a tradução textual resultante corresponda a uma configuração de rede viável e funcional. Como consequência, muitos trabalhos carecem de mecanismos capazes de assegurar que o artefato gerado por LLMs possa ser efetivamente implantado em um ambiente operacional. As abordagens existentes tendem a ser reativas, com ênfase em *troubleshooting* ou na identificação de falhas apenas durante ou após o processo de implantação [Wang et al. 2025]. A principal limitação reside no fato de que o sucesso na tradução textual não implica na viabilidade operacional da rede.

Este trabalho avança o estado da arte ao propor um framework de verificação e validação experimental de configuracões de rede geradas por LLMs. A principal inovação consiste na introdução de um *pipeline* preventivo que integra LLMs a métodos de verificação, permitindo a conferência sintática, a checagem de conformidade até a validação funcional em ambiente emulado. Neste trabalho, validação preventiva refere-se à verificação realizada antes da implantação da configuração, em contraste com abordagens reativas que atuam após a ocorrência de falhas. As principais contribuições deste artigo são: (i) a proposição de um pipeline preventivo multi-estágio para verificação e validação de configurações de rede geradas por LLMs; (ii) a identificação de indícios de degradação no desempenho dos modelos com o aumento do tamanho do lote de tarefas; e (iii) a demonstração de que a validade sintática é insuficiente para garantir a corretude operacional em redes de computadores.

O trabalho se apoia em benchmarks existentes baseados no dataset NetConfEval [Dahlmann et al. 2024], estendendo-os com a inclusão de uma etapa explícita de validação de conformidade e com a incorporação de validação experimental em ambiente emulado, além da avaliação de um novo conjunto de modelos de linguagem de última geração. Paralelamente, o artigo investiga como o volume de tarefas processadas simultaneamente afeta a precisão estrutural das configurações de data plane geradas por LLMs. Ao identificar empiricamente um limiar crítico a partir do qual o desempenho dos modelos se degrada de forma significativa, o estudo evidencia limitações ainda pouco exploradas, com implicações diretas para a automação segura da configuração de redes.

2. Background e Estado da Arte

O uso de Modelos de Linguagem de Grande Porte (LLMs) para a gerência de redes IBN tem apresentado uma evolução acelerada, impulsionada pela capacidade desses modelos em processar requisitos complexos e automatizar a orquestração de rede [Long et al. 2025]. No entanto, a literatura indica que existe uma lacuna entre a capacidade desses modelos de responder questões técnicas e a compreensão real de conceitos de rede [Anwar and Caesar 2024]. Esta seção apresenta as contribuições recentes, fundamentadas nos princípios de Verificação e Validação [IEEE 2012].

Benchmarks e Verificação Sintática: O estágio atual da área é caracterizado pelo surgimento de bases de comparação especializadas, como o dataset NetConfEval [Dahlmann et al. 2024] e o *framework* NetLLMBench [Aykurt et al. 2024]. Enquanto o NetConfEval estabelece um método para quantificar a capacidade das LLMs em traduzir requisitos de rede para modelos estruturados, o NetLLMBench propõe a avaliação de LLMs em tarefas de configuração de rede, estruturado para comparar o desempenho dos modelos em diferentes cenários de gerência.

Verificação de Conformidade e Detecção de Conflitos: No campo da verificação de conformidade, estudos como o Configtrans [Zheng et al. 2024] e as investigações de Hollósi et al. [Hollósi et al. 2024] propõem o uso da consistência de restrições e modelos YANG [Bjorklund 2016] para garantir a integridade estrutural dos dados. Apesar desses avanços, a conformidade estrutural não impede a ocorrência de inconsistências lógicas complexas. Problemas como o sombreamento (*shadowing*) de regras ou rotas recursivas raramente são detectados por validadores de *schema*. Metodologias clássicas de detecção de conflitos, como a proposta pelo algoritmo DETOX [Jesus et al. 2016], oferecem uma base para identificar interdependências lógicas. O framework proposto integra esses princípios para realizar a verificação semântica das políticas geradas.

Abstração Generativa de Intenções: No campo da adaptabilidade, Kou et al. [Kou et al. 2025] introduzem o framework GIA, que utiliza LLMs para criar uma camada de abstração generativa capaz de traduzir requisitos genéricos em modelos de intenção estruturados. A validação do GIA concentra-se na precisão do mapeamento de atributos técnicos, sem contemplar a detecção de conflitos lógicos profundos ou a validação operacional em ambiente de emulação. Essa limitação evidencia a importância de processos de verificação e validação independentes para garantir a funcionalidade efetiva das configurações no *data plane*.

Raciocínio em Múltiplos Passos (*Chain-of-Thought*): Uma evolução em relação aos métodos anteriores, é o uso de estratégias de raciocínio em múltiplos passos, denominadas *Chain-of-Thought* (CoT). O trabalho de Mondal et al. [Mondal et al. 2025] utiliza essa técnica para decompor intenções de usuário vagas ou complexas em sub-etapas técnicas detalhadas antes da geração final. O objetivo é reduzir a ambiguidade na síntese final, forçando o modelo a ”explicar” seu processo lógico antes de entregar o arquivo de configuração. A limitação nesta abordagem reside no fato de que, embora o raciocínio interno do modelo possa parecer coerente textualmente, o resultado final permanece como um artefato não verificado de forma independente. O framework proposto complementa esses avanços ao atuar como um validador independente, assegurando que o produto do raciocínio da LLM seja operacionalmente viável no *data plane*.

Validação Experimental e Limitações: A fronteira final da automação é a executabilidade. Hachimi et al. [Hachimi et al. 2025] propõem validações baseadas em tentativas de implantação (*retry-based deployment*) para regras de redes definidas por software (*Software-Defined Networking* – SDN). Essa abordagem é reativa, o que pode introduzir riscos em redes de produção, uma vez que as falhas são identificadas apenas no momento da implentação.

Validação de Implantação e Garantia de Intenção: No campo da automação fim-a-fim (*end-to-end*), o framework NetIntent [Hossain and Aljoby 2025] utiliza LLMs para realizar a tradução direta de intenções em linguagem natural para regras de fluxo SDN. Entretanto, sua estratégia de validação é reativa, baseando-se em mecanismos de tentativa e erro (*retry-based deployment*) que ocorrem apenas durante a fase de implantação, após a identificação de falhas pelo controlador de rede. Essa abordagem difere do pipeline preventivo proposto. Ele realiza a validação experimental em ambiente de emulação antes da aplicação das configurações na infraestrutura de produção, o que minimiza riscos operacionais de forma antecipada.

Tabela 1. Lacunas identificadas no estado da arte frente ao pipeline proposto.

Trabalho	Sintaxe	Conformidade	Semântica	Validação
NetConfEval	Sim	Parcial	Não	Não
NetLLMBench	Sim	Parcial	Não	Não
Configtrans	Sim	Sim	Não	Não
GIA	Sim	Parcial	Sim	Simulada
NetIntent	Sim	Parcial	Parcial	Reativa
Pipeline proposto	**Sim**	**Sim**	**Sim**	**Preventiva**

A lacuna identificada na literatura é a ausência de um pipeline preventivo que execute uma validação experimental completa em ambiente emulado, como o Mininet [Lantz et al. 2010]. Para contextualizar a contribuição deste trabalho, a Tabela 1 demonstra como o pipeline proposto complementa o estado da arte, comparando-o com benchmarks de referência (NetConfEval [Dahlmann et al. 2024] e NetLLMBench [Aykurt et al. 2024]) e com frameworks de verificação e automação (Configtrans [Zheng et al. 2024], GIA [Kou et al. 2025] e NetIntent [Hossain and Aljoby 2025]).

3. Arquitetura do Pipeline de Verificação e Validação

O framework proposto foi concebido como um pipeline de Verificação e Validação multi-estágio, atuando como uma camada preventiva entre a geração de configurações por LLMs e a infraestrutura de rede. A arquitetura, apresentada na Figura 1, organiza-se em quatro etapas sequenciais que garantem a transição da intenção em linguagem natural para a execução funcional no *dataplane*.

Para ilustrar o funcionamento do pipeline, considere uma intenção de usuário como: *Traffic originating from lyon can reach the subnet 100.0.8.0/24.*[1] O LLM processa esta intenção e gera uma configuração em JSON (formato padrão do dataset NetConfEval [Dahlmann et al. 2024]). Com o arquivo JSON criado, o pipeline inicia as validações e verificações.

[1] Amostra extraída do dataset NetConfEval e que foi aprovada em todas as etapas do pipeline.

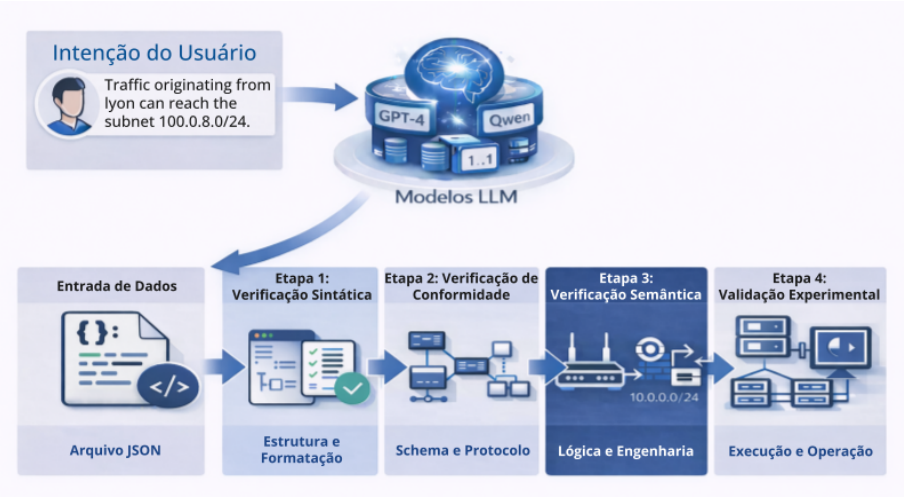

Figura 1. Arquitetura Conceitual do Framework Proposto

A Etapa 1 (Verificação Sintática) isola o objeto estruturado, removendo explicações textuais e delimitadores Markdown da resposta do LLM. Isso assegura que o artefato (o JSON) seja válido para parsing, descartando respostas malformadas.

A Etapa 2 (Verificação de Conformidade) busca a validação do objeto frente ao *schema* YANG [Bjorklund 2016]. Isto quer dizer que os tipos de dados e hierarquias são verificados de acordo com as regras do modelo de dados em questão. No exemplo citado, o framework confirmou se o identificador *lyon* é um *host* válido e que o prefixo *100.0.8.0/24* respeita os padrões de endereçamento IPv4.

O *ground truth* utilizado na Etapa 3 é composto pelo conjunto de respostas de referência do dataset NetConfEval [Dahlmann et al. 2024], elaboradas por especialistas. A Etapa 3 realiza duas funções: (i) comparação semântica com a política de referência, medindo aderência ao benchmark; e (ii) detecção de inconsistências lógicas independentes via DETOX [Jesus et al. 2016], identificando anomalias não capturadas por validadores estruturais. A combinação entre aderência à intenção e ausência de inconsistências define, neste trabalho, um critério operacional de corretude semântica. Reconhece-se que verificadores formais (ex.: Batfish) representam uma extensão futura.

Etapa de Validação Experimental: A etapa final consiste na validação funcional em ambiente de execução simulado. O framework utiliza um orquestrador para traduzir o JSON verificado em comandos de sistema que instanciam a topologia e as tabelas de roteamento no emulador Mininet [Lantz et al. 2010]. O indicador de sucesso nesta fase é o resultado prático da conectividade funcional. No caso do exemplo utilizado, o pipeline realiza testes de transmissão de pacotes (ping) partindo do host *lyon* em direção à sub-rede *100.0.8.0/24*. Esta validação experimental é fundamental para detectar alguma inviabilidade operacional, situação em que a configuração, embora aprovada nas etapas de verificação sintática e de conformidade, falha em estabelecer a comunicação no *dataplane* devido a limitações de adjacência física ou restrições do kernel do sistema operacional que validadores estáticos não conseguem prever.

Exemplo de Instanciação do Pipeline: A Tabela 2 ilustra o fluxo de processamento de uma amostra pelo pipeline, utilizando como exemplo a intenção *Traffic originating from lyon can reach the subnet 100.0.8.0/24*. O exemplo demonstra a progressão

Tabela 2. Exemplo do processamento de uma amostra no pipeline proposto.

Etapa	**Conteúdo / Resultado**	**Status**
Intenção (Input)	”Traffic originating from lyon can reach the subnet 100.0.8.0/24:”	—
Validação Sintática	Código JSON sem erros de formatação ou verbosidade.	Aprovado
Verificação de Conformidade	Conformidade com o Schema NetConf e ausência de conflitos de endereçamento.	Aprovado
Verificação Semântica	Aderência de 100% aos requisitos de conectividade do gabarito (Ground Truth).	Aprovado
Validação no Mininet	`lyon ping -c 3 100.0.8.10` `3 packets transmitted, 3 received, 0% loss`	Sucesso

do artefato desde a recepção da resposta do LLM até a confirmação de sua viabilidade funcional.

Nas etapas iniciais de verificação, o framework atua sobre a integridade do arquivo gerado. Primeiro, assegura-se que a estrutura sintática JSON está correta e limpa de verbosidades típicas de LLMs. Na sequência, a verificação de conformidade valida se os dados respeitam as regras do protocolo e do endereçamento IP. Como terceira etapa, a verificação semântica utiliza o gabarito (*ground truth*) e a detecção de inconsistências para garantir que a política de alcançabilidade para o *host lyon* é logicamente consistente e não sofre interferência de outras regras.

A etapa final do *pipeline* é a validação experimental no Mininet. Como detalhado na Tabela 2, o sucesso nesta fase não depende da estrutura do código, mas da conectividade real medida através da transmissão de pacotes. O resultado de 0% de perda no teste de *ping* confirma que a intenção traduzida é funcional no *data plane*, fechando a lacuna entre a proposta textual do LLM e a operação efetiva da infraestrutura de rede.

4. Metodologia de Avaliação e Resultados

A seleção dos modelos seguiu critérios alinhados à literatura recente, incluindo representatividade arquitetural, alinhamento com os baselines do NetConfEval [Dahlmann et al. 2024] e diversidade de acesso, contemplando tanto modelos proprietários quanto *open-source*.

Os experimentos foram executados em um servidor com CPU Intel Xeon Gold, 128 GB RAM e Ubuntu 22.04. Os modelos foram acessados via APIs, com parâmetros padronizados (T = 0.0, top_p = 1.0). Os prompts seguem o template do NetConfEval, com concatenação de múltiplas intenções para testes de *batch size*. A validação experimental foi conduzida no Mininet, com testes de conectividade via ICMP. O código-fonte do framework está disponível no GitHub[2]. No repositório também estão disponíveis o orquestrador, os scripts de verificação e os templates de prompt de modo que seja possível reproduzir os experimentos realizados neste trabalho.

[2] `https://github.com/cscolomboIA/llm-network-validation-framework`

Metodologia:

A metodologia de avaliação adotada combina a reprodução controlada de *benchmark* do estado da arte com a validação experimental proposta. Para garantir a equivalência científica, o método aplicado faz uso do *dataset* NetConfEval, processando um total de 1.665 tarefas de configuração referentes à Task 1 (*Translating High-Level Requirements to a Formal Specification*) [Dahlmann et al. 2024].

A Task 1 aborda três categorias de políticas com complexidade crescente: *Reachability*, *Waypoint* e *Load-Balancing*. Para os testes de estresse (*batch size*), foram concatenados múltiplos requisitos em um único prompt (variando de 1 a 100 intenções), visando identificar o ponto de saturação da janela de contexto dos modelos.

A escolha pela Task 1 do NetConfEval fundamenta-se na sua proximidade com cenários de implantação, ao exigir a tradução direta de intenções em linguagem natural para especificações formais. Além disso, essa tarefa concentra diversidade de políticas e permite análises com os *baselines* da literatura. A extensão para outras tarefas constitui trabalho futuro.

O diferencial do framework proposto reside na expansão do escopo de avaliação da Task 1 através de etapas em cascata. Diferente da análise estática tradicional, o framework submete as configurações a um *pipeline* onde:

- A Verificação Semântica (Etapa 3) analisa a lógica de engenharia para detectar alucinações conceituais e a omissão de requisitos em lotes de alta densidade (*Batch Dropout*);
- A Validação Experimental (Etapa 4) materializa a configuração no emulador Mininet, onde um orquestrador aplica o saneamento de camadas 2 e 3 para testar a conectividade fim-a-fim.

Análise da Verificação Sintática (Etapas 1 e 2): A Figura 2 apresenta a análise comparativa da acurácia sintática (Etapas 1 e 2 deste trabalho) entre os modelos *baseline* utilizados na pesquisa de [Dahlmann et al. 2024] em 2024 (painel superior) e os modelos selecionados para reproduzirem os experimentos (painel inferior). Os resultados são organizados pelos três níveis de complexidade das políticas e pela variação progressiva do *batch size*.

Os resultados dos *baselines* revelam que, mesmo para modelos como o GPT-4-Turbo, a estabilidade sintática era sensível ao aumento do lote, apresentando quedas acentuadas a partir de 20 tarefas simultâneas, especialmente nas políticas de *Waypoint* e *Load-Balancing*. Modelos de menor escala, como o CodeLlama-13B, demonstravam uma incapacidade estrutural de manter o formato JSON em lotes maiores, aproximando-se de 0% de acurácia nos cenários de maior complexidade.

No painel de resultados experimentais deste trabalho (inferior), nota-se uma evolução significativa na resiliência sintática de alguns modelos mais recentes. O modelo Claude 3.5 Sonnet destaca-se pela estabilidade, mantendo 100% de acurácia em todas as políticas e tamanhos de lote testados. Entretanto, essa característica não é uma regra geral para a geração atual. Modelos como Llama-3.3-70b e Mistral-Large, embora iniciem com bom desempenho, apresentam curvas de degradação similares aos *baselines* de 2024 quando submetidos a lotes superiores a 20 tarefas.

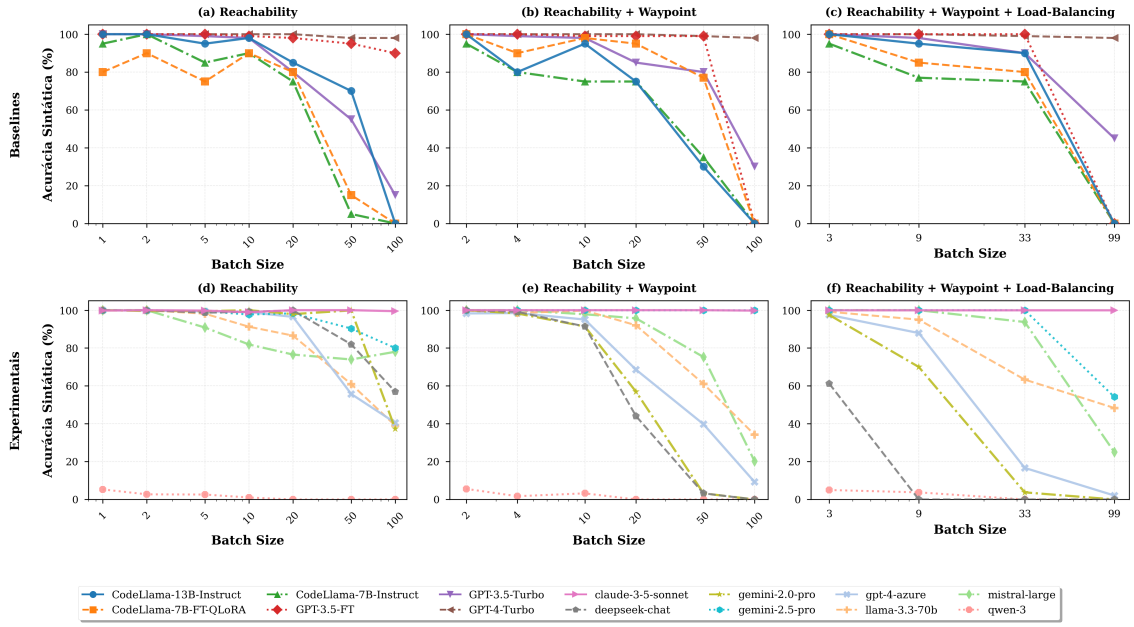

Figura 2. Acurácia Sintática vs Batch Size. Painel superior: modelos baseline da literatura. Painel inferior: modelos experimentais avaliados neste trabalho.

Um ponto crítico identificado na Figura 2(f) é o colapso precoce do modelo DeepSeek-Chat em cenários de *Load-Balancing*, onde a acurácia sintática cai para 0% logo no lote de 9 tarefas. Este fenômeno indica que a alta densidade de requisitos lógicos pode induzir modelos proficientes em código a falhas de formatação.

O desempenho do modelo Qwen-3, que se manteve consistentemente próximo a 0% de acurácia em todos os cenários, atua como uma referência de baixa performance que valida a sensibilidade do ambiente experimental. Esse resultado evidencia que, apesar da evolução geracional das LLMs, a Task 1 do NetConfEval permanece um desafio para arquiteturas que não possuem alinhamento específico com o domínio de redes. A hipótese par justificar este comportamento é o desalinhamento em relação à sintaxe rígida do NetConfEval.

A literatura recente reforça a distinção fundamental entre a capacidade de geração de código e a compreensão real de conceitos de rede. Anwar e Caesar [Anwar and Caesar 2024], ao contextualizarem o benchmark NetConfEval [Dahlmann et al. 2024], argumentam que o sucesso na produção de uma sintaxe estruturada não implica, necessariamente, em um entendimento profundo da lógica de protocolos. Os resultados obtidos com o Qwen-3 e o Llama-3.3-70b corroboram essa tese sob diferentes perspectivas: enquanto o primeiro demonstra incapacidade de aderir ao esquema exigido (falha de domínio), o segundo, apesar de sua competência estrutural, apresenta falhas lógicas e operacionais significativas sob carga de trabalho.

A observação desses comportamentos divergentes reforça a importância do framework proposto. A lacuna identificada entre o ato de ”escrever” a configuração e o ato de ”entender” a topologia justifica a necessidade de um pipeline que transcenda a verificação do artefato textual e valide, efetivamente, sua funcionalidade no *dataplane*. Em síntese, a comparação entre os painéis demonstra que, apesar do amadurecimento

das LLMs na geração de sintaxe estruturada, a vulnerabilidade à saturação da janela de contexto permanece um desafio persistente.

Análise da Verificação Semântica: A Etapa 3 atua como o motor de decisão lógica do *pipeline*, sendo responsável por filtrar inconsistências que a verificação de *schema* (Etapa 2) é incapaz de detectar. Nesta fase, o framework aplica a normalização terminológica e emprega algoritmos de detecção de conflitos e inconsistências lógicas baseados na metodologia DETOX [Jesus et al. 2016], após a representação estruturada das regras de política. Os resultados revelam que uma parcela significativa das falhas operacionais tem origem em erros de interdependência lógica, e não em erros de formatação.

Observou-se que modelos com performance sintática, como o Claude 3.5 Sonnet, apresentam o que se define como alucinação léxica: a geração de nomes de interfaces que divergem da topologia física (ex: usar "GigabitEthernet" em um ambiente que espera "eth"). O processo de normalização semântica da Etapa 3 permitiu resgatar amostras que seriam descartadas por divergências de nomenclatura, mas expôs falhas críticas.

A importância desta etapa é evidenciada ao confrontar a perfeição sintática vista na Figura 2 com os resultados operacionais. A Verificação Semântica detectou que, em muitos casos, o modelo manteve a estrutura JSON impecável, mas falhou ao omitir parâmetros essenciais do gabarito (*ground truth*), fenômeno denominado neste trabalho como *batch dropout*. Este diagnóstico precoce na Etapa 3 impede que configurações logicamente conflitantes avancem para a emulação, economizando recursos computacionais e fornecendo um feedback preciso sobre a falha de raciocínio do modelo.

4.1. Análise da Validação Experimental (Etapa 4):

Uma vez estabelecida a comparação sintática entre os modelos contemporâneos e *baselines*, este tópico avalia se tal precisão estrutural é traduzida em conectividade funcional (como política correspondente, entre outras) no *dataplane*. A Figura 3 apresenta os resultados da validação experimental conduzida no Mininet, correlacionando a taxa de executabilidade operacional com o nível de complexidade das intenções de rede.

A análise do mapa de calor revela que, embora modelos como Claude 3.5 Sonnet tenham alcançado 100% nas fases anteriores, sua executabilidade operacional na Política 1 (*Reachability*) é de 80%. Esta diferença de 20 pontos percentuais aponta o "falso positivo" da verificação sintática isolada: situações em que o JSON é válido, mas a rede falha em estabelecer conectividade. Observa-se um declínio acentuado na performance de quase todos os modelos na transição para a Política 2 (*Waypoint*). O modelo Claude 3.5 Sonnet, por exemplo, sofre uma queda de 60% em sua eficácia operacional. Este cenário indica que a introdução de pontos de passagem obrigatórios impõe uma carga de raciocínio lógico que os LLMs falham em traduzir para uma topologia funcional.

Um achado relevante e contra-intuitivo refere-se ao modelo Llama-3.3-70b. Enquanto a tendência geral é de degradação contínua conforme a complexidade aumenta, este modelo apresentou uma melhora de performance na Política 3 (*Load-Balancing*), alcançando 50% em contraste aos (23,3%) na *Waypoint*. Conforme discutido anteriormente, a maior quantidade de especificações exigidas no *Load-Balancing* parece favorecer a retenção do contexto lógico nesta arquitetura, minimizando alucinações terminológicas comuns em requisitos mais sucintos.

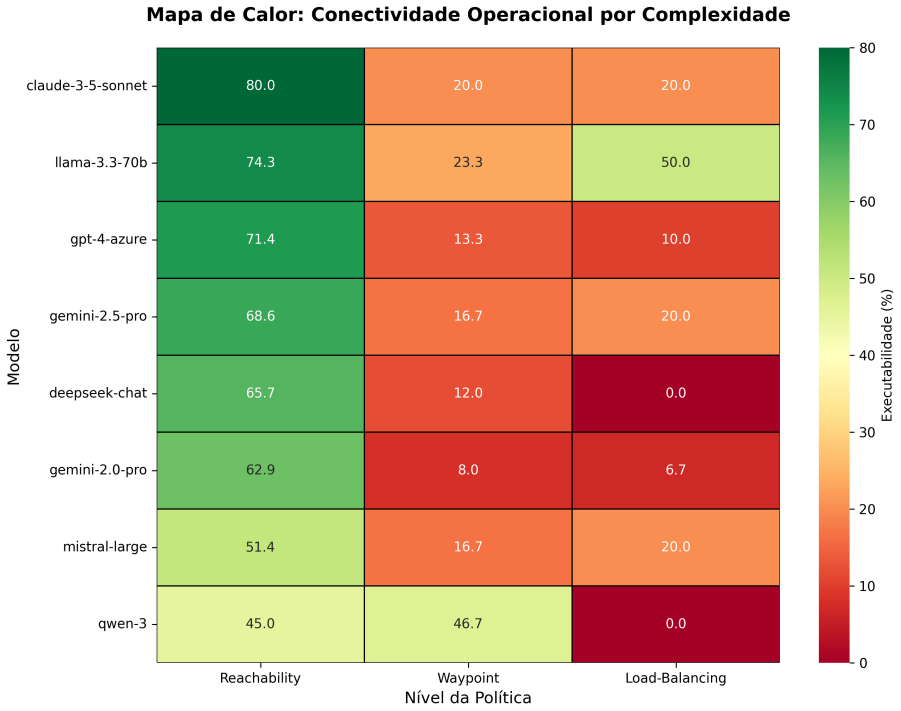

Figura 3. Validação experimental (%) e a complexidade das intenções.

Por fim, a Figura 3 evidencia a inviabilidade operacional de modelos como DeepSeek-Chat e Qwen-3 na Política 3. O registro de 0% de executabilidade confirma que a proficiência em tarefas generalistas de programação não garante a corretude em cenários de infraestrutura complexos. Estes resultados validam a necessidade do pipeline proposto como uma estratégia preventiva que impeça falhas de conectividade alcancem redes de produção.

Tabela 3. Síntese do desempenho dos modelos destaque por política.

Política	**Modelo**	**Destaque**	**Principal Achado**
Reachability	Claude-3.5-Sonnet	80,0%	Maior rigor no seguimento de instruções e baixo ruído sintático.
Waypoint	Llama-3.1-8b	Resiliente	Modelo metódico: gera estruturas consistentes que facilitam a orquestração.
Load-Balancing	Mistral-Large	20,0%	Arquitetura (*Mixture of Experts* - MoE) demonstrou maior robustez para lidar com o número de requisitos.

A análise transversal dos dados revela que a eficácia dos modelos não é uniforme, sendo dependente da natureza da política solicitada e da arquitetura do LLM. O Quadro **??** sintetiza o desempenho operacional e o perfil estatístico dos modelos que se destacaram.

Conforme detalhado no quadro, o modelo Claude-3.5-Sonnet estabelece o teto de performance de 80% para a Política 1 (*Reachability*), o que demonstra um rigor superior no seguimento de instruções negativas (*negative constraints*), o que resulta em baixo ruído sintático. Por outro lado, a Política 2 (*Waypoint*) revela um fenômeno

contra-intuitivo: o Llama-3.1-8b, um modelo de menor escala, superou modelos maiores no quesito resiliência.

Por fim, na Política 3 (*Load-Balancing*), onde o número de requisitos técnicos leva a maioria dos modelos ao colapso, o Mistral-Large sobressaiu-se como o único capaz de manter a viabilidade operacional (20%). Este achado aponta que a arquitetura de Mistura de Especialistas (*Mixture of Experts* - MoE) do Mistral demonstra maior robustez lógica para lidar com a densidade de tokens em cenários de alta complexidade, reafirmando que a precisão em engenharia de redes exige modelos com maior estabilidade entre diferentes domínios de política.

Limitações da Similaridade Textual e Sucesso Operacional: Para aprofundar a compreensão da lacuna entre a geração de código e a corretude funcional, a Figura 4 apresenta a Função de Distribuição Cumulativa Empírica (*Empirical Cumulative Distribution Function* - ECDF) da similaridade textual das configurações geradas pelos modelos em relação ao gabarito (*ground truth*). A análise revela a insuficiência das métricas de similaridade textual para avaliar a real viabilidade operacional das configurações de rede.

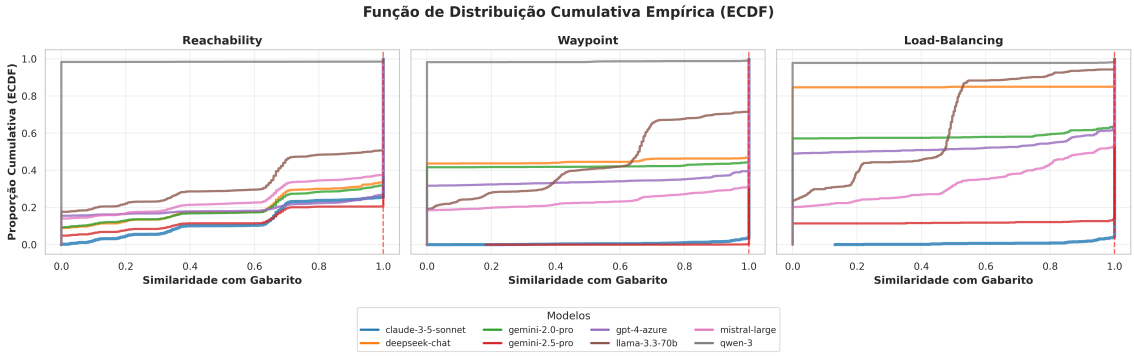

Figura 4. ECDF da similaridade textual com o gabarito para as três políticas.

Conforme ilustrado na Figura 4, o modelo Claude 3.5 Sonnet (linha azul) demonstra alta fidelidade textual em todas as políticas, com a maioria das suas configurações alcançando similaridade próxima a 1.0. Este comportamento, no entanto, contrasta com os resultados do mapa de calor (Figura 3), onde o mesmo modelo apresentou uma taxa de falha de 20% na validação operacional da Política 1. Essa disparidade evidencia que a similaridade textual, embora elevada, esconde falhas sutis de lógica de rede que só são detectáveis em ambiente de execução.

O modelo Llama-3.3-70b (linha marrom), por sua vez, apresenta um comportamento distinto. Sua curva de ECDF na Política 3 (*Load-Balancing*) se eleva em degraus e atinge uma similaridade máxima de aproximadamente 0.6. Isso sugere que o modelo gera configurações estruturalmente corretas, mas com variações substanciais em relação ao gabarito (ex: ordem de chaves, nomenclatura alternativa). Apesar dessa baixa similaridade textual, a validação experimental (Figura 3) indicou um sucesso operacional de 50% para o Llama-3.3-70b na mesma política. Este achado reforça que a corretude de uma rede depende da precisão de tokens críticos e não da fidelidade textual absoluta. Modelos como Qwen-3 (linha cinza) e DeepSeek-Chat (linha laranja) exibem curvas que se mantêm na base do gráfico ou sobem abruptamente para similaridades muito baixas, indicando que a maior parte de suas configurações é textual ou estruturalmente divergente do gabarito.

5. Análise Crítica e Discussão dos Resultados

Esta seção propõe uma interpretação dos dados experimentais através de uma análise estatística e transversal, visando consolidar as evidências sobre a confiabilidade dos LLMs em tarefas de rede.

A Hipótese do *Batch Size* e a Saturação de Atenção: Os resultados evidenciam que o volume de processamento simultâneo é um fator determinante na degradação da corretude. A hipótese do *batch size* sugere um limiar de degradação no desempenho dos modelos com o aumento do tamanho do lote. Esse comportamento foi observado em torno de 20 tarefas por lote. Observou-se que, estatisticamente ($p < 0,05$), a precisão estrutural dos modelos é inversamente proporcional à densidade da janela de contexto. Este fenômeno sugere que, ao ultrapassar este limiar, o LLM sofre uma diluição de foco. Com isso, ele pode confundir ou omitir parâmetros lógicos, invalidando a configuração no *data plane*.

A degradação observada a partir do ponto de inflexão (*batch size* 20) encontra respaldo na literatura sobre arquiteturas de *Transformers*. Conforme demonstrado por Liu et al. [Liu et al. 2024], modelos de linguagem sofrem do fenômeno *lost in the middle*, no qual a capacidade de processamento de informações relevantes diminui drasticamente à medida que a densidade da janela de contexto aumenta. Este efeito de diluição da atenção explica por que o aumento do *batch size* induz o modelo a omitir parâmetros críticos da configuração de rede, resultando no *batch dropout* identificado nesta pesquisa.

Adicionalmente, os resultados operacionais nulos observados em cenários de alta complexidade (Política 3) corroboram a tese de Anwar e Caesar [Anwar and Caesar 2024] sobre a saturação lógica de modelos generalistas. Segundo os autores, quando submetidos a múltiplas restrições técnicas interdependentes, os LLMs atingem um limiar de raciocínio onde a probabilidade de alucinações conceituais supera a precisão sintática. O framework proposto, ao identificar o descolamento entre a corretude do JSON e a falha no *data plane*, fornece evidência empírica para essa saturação, reforçando que o volume de regras é um fator limitante para a automação não supervisionada.

6. Conclusão

A pesquisa demonstrou que a adoção de LLMs para a tradução de intenções de rede exige uma mudança de paradigma: a transição de avaliações puramente textuais para processos de Verificação e Validação. O framework proposto evidenciou que a alta acurácia sintática apresentada por modelos do estado da arte é uma métrica insuficiente, e por vezes enganosa, da capacidade de entrega funcional de uma configuração.

Os resultados experimentais revelaram que a confiabilidade operacional dos LLMs é sensível à densidade do contexto. Identificou-se um ponto de inflexão em torno de 20 tarefas simultâneas, a partir do qual a diluição da atenção induz a erros lógicos que validadores de *schema* convencionais são incapazes de capturar. A análise dos resultados possibilitou a identificação de um hiato entre o código gerado e a conectividade no *data plane*, revelando que mesmo os modelos mais resilientes, como o Claude 3.5 Sonnet, apresentam uma margem de incerteza operacional para infraestruturas críticas sem uma validação preventiva.

A principal contribuição deste artigo reside na arquitetura de Verificação e Validação em etapas, que integra a detecção de inconsistências lógicas à validação ex-

perimental em ambiente emulado. Esta abordagem preenche a lacuna identificada em *benchmarks* anteriores e busca apresentar contribuições para melhorar a eficiência operacional de intenções geradas por LLMs. Como limitação deste estudo, a avaliação está restrita à Task 1 do NetConfEval. Os experimentos foram conduzidos em execução única, sem análise estatística de variância. Além disso, a validação semântica depende parcialmente do benchmark. Aspectos como custo computacional, escalabilidade e integração com ambientes de produção não foram avaliados.

Como trabalhos futuros, pretende-se explorar mecanismos de autorrecuperação (*self-healing*), nos quais os *logs* de erro e as falhas de conectividade identificadas na etapa de validação sejam retroalimentados aos modelos para a correção automatizada das configurações. Pretende-se conduzir um estudo de ablação, visando quantificar a contribuição individual de cada etapa de verificação para a viabilidade operacional final. Planeja-se também investigar o uso de técnicas de Geração Aumentada por Recuperação (*Retrieval-Augmented Generation* - RAG) para aprimorar a capacidade das LLMs em gerar configurações mais precisas. Por fim, planeja-se explorar a integração do pipeline proposto a arquiteturas baseadas em agentes de inteligência artificial, por se entender que este pode ser o caminho natural desta área de pesquisa.

7. Declaração do Uso de ferramentas IA

Os autores utilizaram o ChatGPT para auxílio na revisão gramatical, estruturação estilística e organização textual deste artigo. O conteúdo técnico, a condução dos experimentos, a coleta de dados e a interpretação dos resultados foram realizados e verificados integralmente pelo autor humano, que assume total responsabilidade pela veracidade e integridade das informações apresentadas.

Referências

Anwar, M. and Caesar, M. (2024). Understanding misunderstandings: Evaluating LLMs on networking questions. *ACM SIGCOMM Computer Communication Review*, 54(4):14–24. (Citado nas páginas 2, 3, 8, and 12.)

Aykurt, K., Blenk, A., and Kellerer, W. (2024). NetLLMBench: A benchmark framework for large language models in network configuration tasks. In *Proceedings of the 2024 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 1–6. IEEE. (Citado nas páginas 2, 3, and 4.)

Bhardwaj, A. and Fomina, S. (2025). Probably correct: Rethinking formal verification for LLM-driven systems. In *Proceedings of the 2025 IEEE 12th International Conference on Cyber Security and Cloud Computing (CSCloud)*, pages 82–86. IEEE. (Citado na página 2.)

Bjorklund, M. (2016). The YANG 1.1 data modeling language. RFC 7950, RFC Editor. (Citado nas páginas 3 and 5.)

Dahlmann, T., Gmeiner, J., Krude, J., and Wehrle, K. (2024). NetConfEval: A multi-stage benchmark for network configuration generation with LLMs. *ACM SIGCOMM Computer Communication Review*, 54(2):16–25. (Citado nas páginas 2, 3, 4, 5, 6, 7, and 8.)

Hachimi, A., Cicco, N., Ibrahimi, M., Musumeci, F., and Tornatore, M. (2025). Flow-rule generation for SDN using LLMs with retry-based deployment validation. In *Proceedings of the 21st International Conference on Network and Service Management (CNSM)*, pages 1–5. IFIP. (Citado nas páginas 2 and 4.)

Hollósi, G., Ficzere, D., and Varga, P. (2024). Generative AI for low-level NETCONF configuration in network management based on YANG models. In *Proceedings of the 20th International Conference on Network and Service Management (CNSM)*, pages 1–7. IFIP. (Citado na página 3.)

Hossain, M. and Aljoby, W. (2025). NetIntent: Leveraging large language models for end-to-end intent-based SDN automation. *IEEE Open Journal of the Communications Society*, 6:10512–10541. (Citado nas páginas 2 and 4.)

IEEE (2012). IEEE standard for system, software, and hardware verification and validation. *IEEE Std 1012-2012*. (Citado na página 3.)

Jesus, Y. K. F., Martinello, M., and Zambon, E. (2016). DETOX: Detecção de inconsistências na política de segurança implementada em firewall real. In *Anais do VII Workshop de Testes e Tolerância a Falhas (WTF)*, pages 63–76, Salvador, BA, Brazil. (Citado nas páginas 3, 5, and 9.)

Kou, S., Yang, C., and Gurusamy, M. (2025). GIA: LLM-enabled generative intent abstraction to enhance adaptability for intent-driven networks. *IEEE Transactions on Cognitive Communications and Networking*, 11(2):999–1012. (Citado nas páginas 2, 3, and 4.)

Lantz, B., Heller, B., and McKeown, N. (2010). A network in a laptop: Rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM Workshop on Hot Topics in Networks (HotNets-IX)*, pages 1–6, New York, NY, USA. ACM. (Citado nas páginas 4 and 5.)

Liu, N. F., Lin, K., and Chen, e. (2024). Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173. (Citado na página 12.)

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. (Citado na página 3.)

Mondal, R., Bjorner, N., Millstein, T., Tang, A., and Varghese, G. (2025). Tackling ambiguity in user intent for LLM-based network configuration synthesis. In *Proceedings of the 24th ACM Workshop on Hot Topics in Networks (HotNets)*, pages 176–183. ACM. (Citado na página 3.)

Tu, N., Nam, S., and Hong, J. W.-K. (2024). Intent-based network configuration using large language models. *International Journal of Network Management*, 35(1). (Citado na página 2.)

Wang, Z., Cornacchia, A., Galante, F., Centofanti, C., Sacco, A., and Jiang, D. (2025). Towards a playground to democratize experimentation and benchmarking of AI agents for network troubleshooting. In *Proceedings of the 1st Workshop on Next-Generation Network Observability*, pages 1–3. ACM. (Citado na página 2.)

Zheng, N., Li, F., Li, Z., Yang, Y., Hao, Y., Liu, C., and Wang, X. (2024). Configtrans: Network configuration translation based on large language models and constraint solving. In *Proceedings of the 32nd IEEE International Conference on Network Protocols (ICNP)*, pages 1–12. IEEE. (Citado nas páginas 2, 3, and 4.)

Zhou, Y., Hsieh, K., Mani, S., Kandula, S., and Liu, Z. (2025). MeshAgent: Enabling reliable network management with large language models. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 9(3):1–36. (Citado na página 2.)