PSLM_NetConfig: Uma Abordagem Eficiente em Recursos para Geração de Configurações de Rede a partir de Intenções em Linguagem Natural

Luis F. Solis Navarro[1], **Rómulo W. C. Bustincio**[1], **and Carlos A. Astudillo**[1]

[1] Instituto de Computação – Universidade Estadual de Campinas (UNICAMP)
Av. Albert Einstein, 1251 – Cidade Universitária Zeferino Vaz
13083-852 – Campinas – SP – Brazil

l2146@dac.unicamp.br, {romulo.bustincio, castudillo}@unicamp.br

Abstract. *Intent-driven self-configuration has emerged as a central capability within the Zero-touch Network and Service Management (ZSM) paradigm in modern complex networks. Although SLM-based approaches adapted with LoRA have shown strong results in translating natural-language intents into network configurations, they still rely on adaptation processes that remain computational costly. To address this limitation, this work proposes PSLM_NetConfig, an approach that combines Prompt Tuning with structured prompting techniques on top of the small language model, while keeping the model weights frozen and updating only 0.003% of its parameters. The best configuration, based on Meta Prompting, achieved 53.3% syntactic accuracy and 70.0% goal accuracy, approaching the performance of a fine-tuned model baseline while reducing training time by approximately* ***72%*** *and memory usage by* ***44%****. These results indicate that the combination of Prompt Tuning and structured prompting constitutes an efficient and computationally lightweight alternative to fine-tuning for network configuration generation in resource-constrained scenarios.*

Resumo. *A autoconfiguração orientada por intenções emergiu como uma capacidade central dentro do paradigma de Gerenciamento Zero-touch Network and Service Management (ZSM). Embora abordagens baseadas em Small Language Models (SLM) adaptadas com LoRA tenham apresentado resultados expressivos na tradução de intenções em linguagem natural para configurações de rede, elas ainda dependem de processos de adaptação que permanecem computacionalmente custosos em termos de tempo, memória e número de parâmetros treináveis. Para enfrentar essa limitação, este trabalho propõe o* ***PSLM_NetConfig****, uma abordagem que combina Prompt Tuning com técnicas estruturadas de engenharia de prompt sobre um modelo de linguagem pequeno fundacional, preservando os pesos e atualizando apenas* ***0,003%*** *de seus parâmetros. A melhor configuração, baseada em Meta Prompting, alcançou* ***53,3%*** *de acurácia sintática e* ***70,0%*** *de acurácia de objetivo, aproximando-se do desempenho do mesmo modelo ajustado por fine-tuning, com redução de aproximadamente* ***72%*** *no tempo de treinamento e* ***44%*** *no uso de memória. Esses resultados indicam que a combinação entre Prompt Tuning e prompting estruturado constitui uma alternativa eficiente e computacionalmente leve ao*

fine-tuning para a geração de configurações de rede em cenários com restrições de recursos.

1. Introduccion

A configuração e o gerenciamento de redes modernas tornaram-se tarefas de crescente complexidade operacional [Liyanage et al. 2022]. A proliferação de tecnologias como redes definidas por software, virtualização de funções de rede e infraestruturas 5G/6G introduziu uma quantidade massiva de parâmetros interdependentes que, em sua maioria, ainda dependem de intervenção humana especializada para sua correta configuração [Pang et al. 2020]. Essa dependência de expertise técnica aprofundada compromete a escalabilidade, aumenta a probabilidade de erros e dificulta a adoção em larga escala dos princípios de automação preconizados pelo framework *Zero-touch Network and Service Management* (ZSM) da ETSI [ETSI ISG ZSM 2018]. Nesse cenário, os Grandes Modelos de Linguagem (LLMs) emergem como uma alternativa promissora para apoiar especialistas em tarefas que exigem acesso rápido a conhecimento técnico e específico de domínio [de Lima et al. 2024], pois demonstram capacidade de interpretar intenções expressas em linguagem natural e traduzi-las em instruções operacionais, estabelecendo uma ponte entre o nível de abstração dos operadores de rede e a sintaxe de baixo nível dos dispositivos [Bimo et al. 2025].

Entretanto, a capacidade de generalização dos LLMs sobre linguagem natural não se traduz automaticamente em precisão técnica em domínios especializados [Huang et al. 2024]. Quando aplicados a tarefas que demandam conhecimento altamente específico, como a geração de comandos Cisco IOS [Cisco Systems 2006] para configuração de roteadores, esses modelos estão sujeitos ao fenômeno das *alucinações*, produzindo saídas linguisticamente coerentes, mas factualmente incorretas, sintaticamente inválidas ou semanticamente dissociadas do requisito original [Huang et al. 2025]. Esse problema decorre, em grande medida, da limitada cobertura de conhecimento de domínio nos corpora de pré-treinamento, o que leva o modelo a preencher lacunas com padrões estatisticamente plausíveis, porém tecnicamente equivocados, sobretudo diante de terminologia rara ou configurações *vendor-specific* [Tonmoy et al. 2024].

Para mitigar esse problema, a estratégia mais direta consiste em submeter o modelo a um processo de *fine-tuning* supervisionado sobre dados específicos do domínio-alvo, permitindo-lhe internalizar padrões sintáticos e semânticos e, assim, reduzir substancialmente a taxa de alucinações em tarefas *downstream* [Parthasarathy et al. 2024]. No entanto, o *fine-tuning* completo de um LLM exige a atualização de bilhões de parâmetros, demandando infraestrutura de GPU de alto desempenho, longos períodos de treinamento e grandes volumes de dados anotados, recursos frequentemente indisponíveis em ambientes de borda ou em contextos com restrições computacionais [Dettmers et al. 2023]. Para contornar esse custo proibitivo, surgiram técnicas de *Parameter-Efficient Fine-Tuning* (PEFT), entre as quais se destaca o LoRA [Hu et al. 2022], que congela os pesos pré-treinados do modelo e insere matrizes treináveis de baixo ranque nas camadas de atenção, reduzindo significativamente o número de parâmetros ajustáveis e viabilizando a especialização de modelos de grande porte em hardware mais acessível, sem comprometer o conhecimento geral do modelo base [XU et al. 2026].

Nessa direção, o trabalho SLM_NetConfig [Lira et al. 2025] demonstra que o

ajuste com LoRA sobre o modelo Zephyr-7B-β constitui uma estratégia eficaz para especializar pequenos modelos de linguagem na tradução de requisitos em linguagem natural para configurações Cisco IOS, alcançando desempenho sintático e semântico superior ao de diferentes *baselines*, inclusive abordagens baseadas em *few-shot prompting*. Apesar desses resultados promissores, a abordagem ainda apresenta limitações importantes que restringem sua adoção em cenários com severas restrições de recursos, como dispositivos de borda, nos quais mesmo a atualização de uma fração reduzida dos parâmetros pode representar um custo significativo em memória, processamento e tempo de treinamento. Além disso, o método depende de um pipeline relativamente elaborado de engenharia de dados para construir o conjunto de treinamento e, embora utilize estratégias de *few-shot learning* para melhorar a qualidade das saídas, ainda deixa em aberto a exploração mais sistemática de técnicas recentes de *prompt engineering*, capazes de orientar melhor o comportamento do modelo, aumentar a consistência estrutural das respostas e reduzir a necessidade de adaptação paramétrica.

Diante dessas lacunas, técnicas de adaptação leve e de engenharia de prompts surgem como alternativas promissoras. O *Prompt Tuning* [Lester et al. 2021], por exemplo, permite adaptar o comportamento do modelo por meio do treinamento de um pequeno conjunto de *tokens* virtuais adicionados à entrada, mantendo congelados todos os parâmetros do modelo base e reduzindo drasticamente o custo computacional. De forma complementar, estratégias de *prompt engineering* como *Chain-of-Thought* (CoT) [Wei et al. 2022], *ReAct* [Yao et al. 2023] e *Meta Prompting* [Zhang 2024] oferecem mecanismos para estruturar melhor o processo de geração e potencialmente explorar com maior eficiência o conhecimento já presente no modelo pré-treinado.

A partir desse contexto, este trabalho propõe o **PSLM_NetConfig**, uma abordagem que combina *Prompt Tuning*, no contexto de PEFT, com técnicas recentes de *Prompt Engineering*, aplicadas ao modelo base Zephyr-7B-β e avaliadas sobre o mesmo conjunto de referência de 70 requisitos utilizado por Lira et al. [Lira et al. 2025]. O objetivo é investigar se o treinamento de apenas uma pequena fração de parâmetros adicionados ao modelo, representados por *tokens* virtuais, aliado ao uso de estratégias estruturadas de prompting, pode produzir desempenho equivalente ou superior ao de abordagens baseadas em ajuste fino mais custoso. Com isso, busca-se verificar se adaptações leves são suficientes para explorar de forma mais eficiente o conhecimento já presente no modelo pré-treinado para a tarefa de geração de configurações de rede.

2. Trabalhos Relacionados

A utilização de LLMs para suportar o paradigma Zero-touch Network and Service Management (ZSM) tem avançado rapidamente e diferentes trabalhos já demonstram que intents em linguagem natural podem ser traduzidas em configurações executáveis de rede [El Rajab et al. 2024]. Contudo, a maioria dessas soluções depende de modelos de grande porte ou de processos de fine-tuning custosos, o que limita sua adoção em ambientes de borda ou infraestruturas com GPUs restritas. Portanto, emerge a necessidade de abordagens que preservem a acurácia sintática e semântica, mas que reduzam ao mínimo o número de parâmetros treináveis e o consumo de recursos durante treinamento e inferência.

Soluções como NETBUDDY [Wang et al. 2023] mostram que modelos como o

GPT-4 podem gerar configurações P4 e BGP a partir de requisitos de alto nível, dividindo o processo em etapas e explorando engenharia de prompts para reduzir erros de tradução. E LLM-NetConfig [Lira et al. 2024] demonstra que um LLM local baseado no Zephyr-7B [Hugging Face H4 Team 2023] é capaz de traduzir intents em configurações Cisco IOS, integrando um verificador para checagem sintática e de objetivo, e eliminando a dependência de plataformas em nuvem. Mas ambas as propostas continuam ancoradas em modelos relativamente grandes e não exploram sistematicamente técnicas de adaptação leve em parâmetros, o que torna sua replicação em dispositivos de borda ainda desafiadora.

Na direção da eficiência, **SLM_NetConfig** introduz um framework baseado em Small Language Models (SLMs) finamente ajustados com técnicas de PEFT, combinando quantização em 4 bits e LoRA sobre o mesmo modelo Zephyr-7B-β. E os resultados mostram ganhos expressivos de acurácia sintática e de objetivo em relação ao LLM-NetConfig, ao mesmo tempo em que reduzem a latência de tradução e produzem configurações mais concisas. Mas esse ganho vem ao custo de um pipeline de engenharia de dados sofisticado e de um processo de fine-tuning supervisionado, que ainda exigem infraestrutura dedicada e tempo de treinamento não trivial. Portanto, permanece aberta a questão de se abordagens ainda mais leves podem atingir desempenho comparável sem atualizar pesos internos do modelo base.

Em paralelo, a literatura de PEFT demonstra que é possível especializar LLMs com uma fração dos parâmetros treináveis. LoRA e variantes reduzem de uma a duas ordens de grandeza o número de parâmetros atualizados, preservando grande parte do desempenho em tarefas de domínio específico [Hu et al. 2022]. E o Prompt Tuning mostra que *soft prompts* aprendidos podem igualar, em modelos grandes, o desempenho do fine-tuning completo, mantendo o modelo congelado e treinando apenas um pequeno conjunto de vetores de entrada [Lester et al. 2021]. Mas essas técnicas têm sido pouco exploradas em sistemas de autoconfiguração de redes, em que prevalecem arquiteturas baseadas em fine-tuning com LoRA ou uso direto de LLMs genéricos. Portanto, há espaço para investigar se o ajuste exclusivo de tokens virtuais já é suficiente para internalizar padrões Cisco IOS sem comprometer a eficiência.

Ao mesmo tempo, avanços em *prompt engineering* oferecem mecanismos complementares para controlar o raciocínio sem alterar parâmetros. *Chain-of-Thought* (CoT) explicita cadeias intermediárias de raciocínio e melhora o desempenho em tarefas complexas de raciocínio algébrico e lógico [Wei et al. 2022], enquanto ReAct intercala raciocínio e ações sobre o ambiente para reduzir alucinações em tarefas de busca e verificação factual. Mais recentemente, o *Meta Prompting* propõe estruturas de alto nível que decompõem tarefas em subtarefas coordenadas por um "condutor", permitindo que um único modelo atue como múltiplos especialistas sem necessidade de treinamento adicional [Suzgun and Kalai 2024]. Contudo, trabalhos em redes empregam sobretudo templates estáticos e poucos exemplos, sem explorar de modo sistemático essas estratégias de raciocínio estruturado.

Diante desse panorama, este trabalho posiciona-se como uma alternativa que combina adaptação leve via **Prompt Tuning** com **técnicas de engenharia de prompts**. As abordagens existentes mostram que fine-tuning eficiente em parâmetros e prompts bem projetados são cruciais para reduzir alucinações e melhorar a aderência sintática de

configurações geradas. Mas nenhuma delas avalia, sob as mesmas condições de modelo base e conjunto de requisitos Cisco IOS, se o treinamento de **apenas um conjunto reduzido de tokens virtuais**, aliado a **CoT, ReAct o Meta Prompting**, pode igualar ou superar o desempenho de um SLM finamente ajustado com LoRA. Portanto, o **PSLM_NetConfig** busca preencher essa lacuna, investigando se adaptações estritamente leves são suficientes para explorar o conhecimento já presente no Zephyr-7B-β com menor custo computacional.

3. Metodologia

Esta seção detalha os dados de treinamento (Prompt Tuning) e avaliação do PSLM_NetConfig. Conforme a metodologia de [Lira et al. 2025], mantemos a separação estrita entre ambos para garantir que os testes representem cenários de configuração inéditos.

3.1. Dados de Treinamento

Para a adaptação via *Prompt Tuning* com PEFT, utilizamos um conjunto supervisionado de 5 097 pares pergunta–configuração[1], em que cada entrada descreve uma intenção de configuração de rede em linguagem natural e cada saída corresponde à sequência de comandos Cisco IOS necessária para implementá-la. Esses pares foram derivados de requisitos extraídos da documentação técnica Cisco IOS XE Gibraltar 16 [Cisco Systems 2024] e enriquecidos por um processo de reformulação textual que converteu os enunciados originais em estruturas interrogativas, aproximando o treinamento de cenários reais em que administradores de rede expressam solicitações como perguntas. Além disso, houve padronização da terminologia e verificação da coerência entre cada intenção e a respectiva configuração, favorecendo maior consistência sintática e semântica durante o treinamento. As entradas em linguagem natural apresentam formulações variadas, com múltiplos padrões linguísticos, o que amplia a diversidade semântica observada pelo modelo, enquanto as saídas combinam trechos explicativos com blocos de comandos Cisco IOS, mantendo uma estrutura instrucional estável. O processo de treinamento preserva o mesmo formato instrucional empregado no conjunto original, garantindo correspondência direta entre solicitação e configuração gerada e permitindo comparabilidade estrita com o *fine-tuning* via LoRA, uma vez que ambas as abordagens são treinadas sobre os mesmos pares pergunta–configuração e avaliadas pelas mesmas métricas [Lira et al. 2025].

3.2. Treinamento do Modelo

O treinamento do modelo foi realizado em uma GPU NVIDIA com 16 GB de VRAM. Para viabilizar a adaptação do modelo dentro das restrições de memória disponíveis, adotou-se quantização em 4 bits no formato NF4 [Dettmers et al. 2023], o que permitiu reduzir significativamente o consumo de memória sem comprometer a qualidade das representações aprendidas. Cada exemplo de treinamento é estruturado como um par *(intenção, configuração)*, em que a entrada expressa, em linguagem natural, uma solicitação de configuração de rede, e a saída correspondente apresenta os comandos Cisco IOS necessários para sua implementação. As sequências de entrada foram limitadas a 50 tokens, o que, considerando o vocabulário do modelo base de 32.000 tokens

[1]Os dados e o código-fonte utilizados neste trabalho estão disponíveis no repositório de GitHub: **`https://github.com/luisfernandosolis/PSML_NetConfig`**.

e uma dimensão de *embedding* de 4.096, resulta em representações de entrada com dimensão 50×4.096. Um exemplo representativo desse formato é apresentado a seguir:

- **Pergunta:** *"How do I enable OSPF routing on all interfaces?"*
- **Comando de configuração:** `router ospf 1 network 192.168.1.0 0.0.0.255 area 0`

A Figura 1 mostra a perda ao longo de 5 épocas. Cada época corresponde a uma passagem completa pelos dados, enquanto cada *step* representa uma atualização dos pesos por *batch*. Assim, os ∼3000 *steps* correspondem ao total de atualizações realizadas ao longo das 5 épocas.

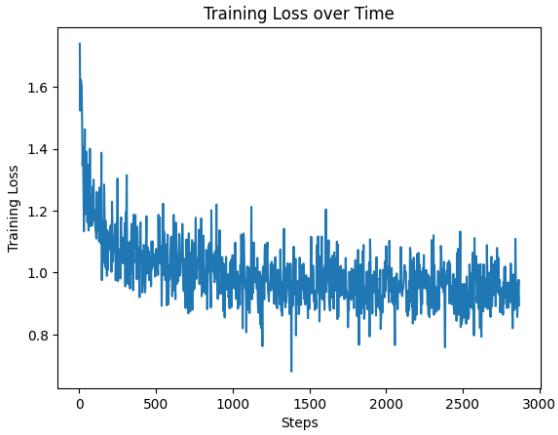

Figura 1. Perda de treinamento ao longo de 5 épocas.

3.3. Conjunto de Avaliação

Para a avaliação do **PSLM_NetConfig**, adotamos um único conjunto de teste com 70 requisitos de configuração de rede. Essa decisão foi tomada porque a base disponibilizada com os valores de referência necessários para comparação continha apenas 70 registros acessíveis, o que impossibilitou a realização de uma análise comparativa consistente sobre a totalidade das instâncias originalmente mencionadas no protocolo de avaliação. Dessa forma, todas as métricas consideradas neste trabalho, incluindo acurácia sintática, acurácia de objetivo, tempo de tradução e pontuação de complexidade, foram calculadas sobre esse mesmo subconjunto. Essa escolha assegura uniformidade metodológica entre as avaliações e permite analisar o desempenho do modelo sobre um conjunto único, controlado e comparável. O subconjunto de avaliação reúne 70 requisitos distribuídos entre seis categorias de configuração, com predominância de Propriedades de Configuração (41,4%), seguidas por Listas de Acesso (22,9%) e Encaminhamento de Pacotes (20,0%). Em menor proporção, aparecem Protocolos de Roteamento Topológico (10,0%), bem como Roteamento e Tunelamento, ambas com 2,9% dos casos. Essa distribuição heterogênea de categorias contribui para uma avaliação abrangente, cobrindo tanto cenários de configuração frequentes quanto casos menos comuns, o que permite estimar a capacidade de generalização do modelo proposto em contextos reais de rede.

3.4. Métricas de Avaliação

Para assegurar comparabilidade metodológica com o trabalho de referência [Lira et al. 2025], o **PSLM_NetConfig** foi avaliado com base no mesmo

conjunto de métricas adotado no protocolo experimental original, abrangendo quatro dimensões complementares: acurácia sintática, acurácia de objetivo, tempo de tradução e complexidade da configuração gerada. Neste trabalho, contudo, todas essas métricas foram calculadas sobre um único subconjunto de 70 requisitos, selecionado a partir da base de avaliação disponível, de modo a manter consistência entre as comparações e uniformidade na análise de desempenho.

3.4.1. Acurácia Sintática

A Acurácia Sintática mede se a configuração gerada a partir de um requisito de intenção está em conformidade com a sintaxe esperada de comandos Cisco IOS. Essa métrica é importante porque a presença de erros sintáticos compromete diretamente a aplicabilidade da configuração, mesmo quando a intenção geral do comando parece correta.

A avaliação é realizada em dois níveis. No primeiro, cada linha de configuração l_i é analisada individualmente pela função valid(l_i), apresentada na Equação 1, que atribui valor 1 para linhas válidas, 0 para linhas incompletas ou ambíguas, e -1 para linhas inválidas.

$$\text{valid}(l_i) = \begin{cases} 1 & \text{se } l_i \text{ corresponde a um comando de configuração conhecido} \\ 0 & \text{se } l_i \text{ está incompleta ou contém parâmetros incomuns} \\ -1 & \text{se } l_i \text{ é inválida} \end{cases} \tag{1}$$

No segundo nível, a Acurácia Sintática da configuração completa $C = \{l_1, l_2, \ldots, l_n\}$ é obtida pela agregação das avaliações de todas as linhas, conforme a Equação 2. O critério é conservador: a configuração recebe pontuação 1 apenas quando todas as linhas são válidas, 0 quando pelo menos uma linha é incompleta, e -1 quando existe ao menos uma linha inválida.

$$\text{Syntax Accuracy} = \begin{cases} 1 & \text{se } \forall\, l_i,\ \text{valid}(l_i) = 1 \\ 0 & \text{se } \exists\, l_i : \text{valid}(l_i) = 0 \\ -1 & \text{se } \exists\, l_i : \text{valid}(l_i) = -1 \end{cases} \tag{2}$$

Na prática, como este trabalho busca uma pipeline automática tanto para geração quanto para avaliação, adotou-se a estratégia *LLM-as-a-Judge* [Zheng et al. 2023] para verificar a validade sintática das saídas. Em particular, utilizamos o mesmo modelo reportado no protocolo experimental de referência, o Gpt-4o-mini, para atuar como juiz na análise linha a linha das configurações geradas. Essa decisão preserva a coerência metodológica com os experimentos anteriores e permite automatizar a avaliação mantendo um critério consistente entre os diferentes resultados analisados.

3.4.2. Acurácia de Objetivo

A Acurácia de Objetivo mede em que grau a configuração gerada C atende ao requisito de intenção de rede R. Essa métrica é fundamental porque uma configuração pode ser

sintaticamente correta e, ainda assim, falhar em cumprir o objetivo operacional solicitado.

Sua definição é apresentada na Equação 3. A pontuação é 1 quando C satisfaz completamente R, 0 quando o atendimento é apenas parcial, e -1 quando a configuração não satisfaz o requisito.

$$\text{Goal Accuracy} = \begin{cases} 1 & \text{se } C \text{ satisfaz completamente } R \\ 0 & \text{se } C \text{ satisfaz parcialmente } R \\ -1 & \text{se } C \text{ não satisfaz } R \end{cases} \tag{3}$$

Na prática, seguindo o mesmo procedimento adotado na métrica anterior, a avaliação foi automatizada por meio da estratégia *LLM-as-a-Judge*. Para isso, utilizamos o Gpt-4o-mini como juiz semântico, responsável por verificar se a configuração produzida atende ao objetivo expresso no requisito em linguagem natural, independentemente de pequenas variações de forma ou ordenação dos comandos. A distinção entre atendimento parcial e falha completa é especialmente relevante, pois permite capturar com maior fidelidade o grau real de aderência da saída gerada ao objetivo solicitado.

3.4.3. Tempo de Tradução

O Tempo de Tradução mede o intervalo necessário para converter um requisito de intenção em uma configuração executável. Essa métrica é relevante no contexto deste trabalho porque, além da correção da saída, interessa avaliar a viabilidade prática do modelo em cenários de automação de redes, nos quais menor latência de resposta favorece uso operacional mais eficiente.

Seguindo o protocolo adotado no trabalho de referência, o tempo total de tradução é definido pela soma do tempo de processamento do requisito de configuração, T_{pci}, com o tempo de geração do comando correspondente, T_{gcc}, conforme a Equação 4. Em seguida, esse valor é normalizado por escalonamento min-max sobre o conjunto de teste, produzindo norm_total_dura_time, o que permite comparar de forma mais justa diferentes instâncias e modelos, como mostrado na Equação 5.

$$\text{total\_dura\_time} = \frac{T_{pci} + T_{gcc}}{60} \tag{4}$$

$$\text{norm\_total\_dura\_time} = \frac{\text{total\_dura\_time} - \min(\text{total\_dura\_time})}{\max(\text{total\_dura\_time}) - \min(\text{total\_dura\_time})} \tag{5}$$

No **PSLM_NetConfig**, T_{pci} corresponde ao tempo de inferência necessário para processar a intenção de entrada com o modelo base adaptado por *Prompt Tuning*, enquanto T_{gcc} corresponde ao tempo gasto para gerar a configuração final. Dessa forma, a métrica captura não apenas a velocidade de resposta do modelo, mas também sua adequação a cenários com restrições de tempo e recursos, aspecto especialmente importante em ambientes de operação local e automação assistida.

3.4.4. Pontuação de Complexidade

A Pontuação de Complexidade resume, em uma única métrica, o custo operacional associado à geração da configuração. Sua relevância está em combinar dois aspectos práticos do processo de tradução: o tempo gasto na geração e o tamanho total da configuração produzida.

Seguindo o protocolo adotado no trabalho de referência, a complexidade é definida como a média entre o comprimento normalizado da configuração, norm_total_len, e o tempo de tradução normalizado, norm_total_dura_time, conforme a Equação 6. O comprimento normalizado é obtido a partir de total_len, definido como a soma entre o tamanho da intenção de entrada, trans_len, e o tamanho da configuração gerada, config_len, como mostrado nas Equações 7 e 8.

$$\text{complexity\_score} = \frac{\text{norm\_total\_len} + \text{norm\_total\_dura\_time}}{2} \tag{6}$$

$$\text{norm\_total\_len} = \frac{\text{total\_len} - \min(\text{total\_len})}{\max(\text{total\_len}) - \min(\text{total\_len})} \tag{7}$$

$$\text{total\_len} = \text{trans\_len} + \text{config\_len} \tag{8}$$

Com isso, a métrica assume valores no intervalo $[0, 1]$, em que valores menores indicam saídas mais concisas e mais rápidas de gerar, característica desejável em sistemas de configuração automática.

4. Discussão dos Resultados

4.1. Acurácia Sintática

A Figura 2 (no lado esquerdo) mostra que as variantes do **PSLM_NetConfig** alcançam desempenho sintático comparável ao baseline LLM_NetConfig [Lira et al. 2024], e em alguns casos ligeiramente superior, na avaliação dos 70 requisitos do conjunto de teste. Enquanto o LLM-NetConfig obteve 41,4% de respostas corretas e 46,5% de saídas incorretas, as abordagens propostas apresentaram ganhos graduais na proporção de acertos. O **PSLM_NetConfig** (CoT) alcançou 48,0% de respostas corretas, o **PSLM_NetConfig** (ReAct) chegou a 51,3% e o **PSLM_NetConfig** (Meta) obteve 53,3%. Em paralelo, a taxa de respostas incorretas foi reduzida para 17,3%, 12,7% e 8,7%, respectivamente, indicando que o uso de *Prompt Tuning* combinado com estratégias de prompting estruturado contribui para melhorar a validade sintática das configurações geradas por meio da otimização de *tokens* virtuais treináveis adicionados à entrada do modelo.

A comparação entre as três variantes do **PSLM_NetConfig** evidencia que, à medida que a estrutura do prompt se torna mais explícita, o desempenho do modelo melhora de forma consistente. O CoT já produz ganhos ao organizar o raciocínio em etapas, o ReAct amplia essa melhora ao combinar raciocínio e ação, e o *Meta Prompting* apresenta o melhor resultado ao impor uma estrutura de saída mais rígida e informativa. Em conjunto, esses resultados reforçam que a qualidade da engenharia de prompt exerce papel central na redução de alucinações sintáticas e na geração de configurações Cisco IOS mais válidas e confiáveis.

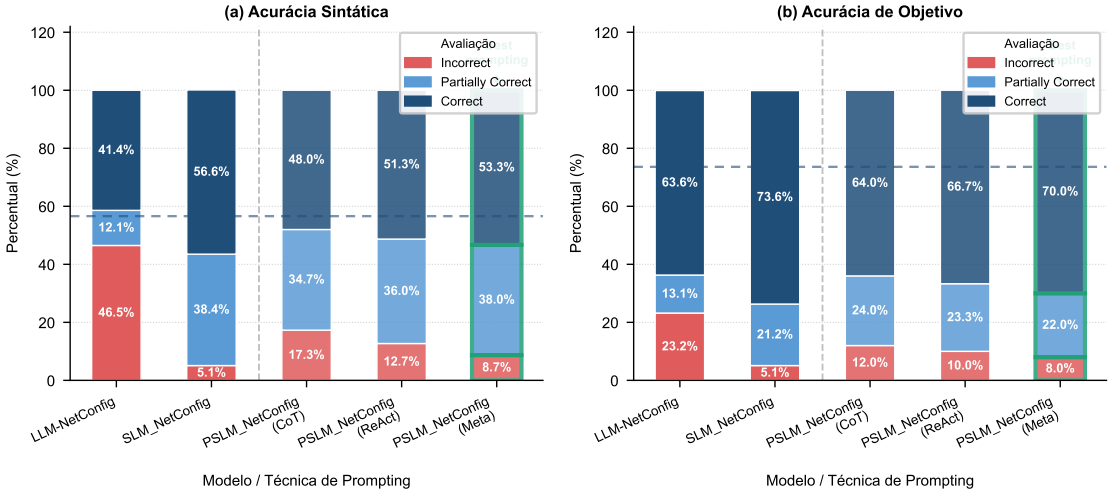

Figura 2. Acurácia sintática (esquerda) e acurácia de objetivo (direita) por modelo. As barras empilhadas mostram a proporção de respostas corretas, parcialmente corretas e incorretas nos 70 requisitos avaliados, enquanto as linhas tracejadas indicam os valores de referência do SLM_NetConfig: 56,6% para acurácia sintática e 73,6% para acurácia de objetivo.

4.2. Acurácia de Objetivo

A Figura 2 (lado direito) mostra que o **PSLM_NetConfig** com Meta Prompting alcança desempenho semântico próximo ao do SLM_NetConfig (70,0% vs. 73,6% de corretos; 8,0% vs. 5,1% de falhas) e superior ao LLM-NetConfig (63,6% de corretos, 23,2% de incorretos). As variantes apresentam incrementos graduais (CoT: 64,0%; ReAct: 66,7%; Meta: 70,0%), sugerindo que prompts mais estruturados favorecem tanto a forma dos comandos quanto a aderência ao objetivo em linguagem natural.

4.3. Pontuação de Complexidade

Na parte esquerda da Figura 3 mostra que o **PSLM_NetConfig** mantém um perfil de complexidade intermediário, refletindo um compromisso prático entre concisão da saída e baixo custo de adaptação do modelo. Embora os valores permaneçam acima dos reportados para o **SLM_NetConfig** fine-tunado, a abordagem proposta opera sob uma restrição metodológica distinta, pois se apoia em *Prompt Tuning* e técnicas de prompting estruturado, sem atualização completa dos parâmetros internos.

Sob essa perspectiva, os resultados indicam que o modelo proposto consegue reproduzir parcialmente o comportamento do baseline fine-tunado com uma estratégia significativamente mais leve em termos de treino e uso de recursos. Isso reforça a viabilidade de métodos de adaptação eficiente como alternativa prática para cenários em que custo computacional, simplicidade de implantação e preservação do modelo base são fatores centrais.

4.4. Tempo de Tradução

No lado direito da Figura 3 mostra que o **PSLM_NetConfig** mantém um perfil de tempo de tradução competitivo ao longo dos requisitos avaliados, com comportamento mais estável que o observado nos modelos de referência. Em particular, a abordagem proposta

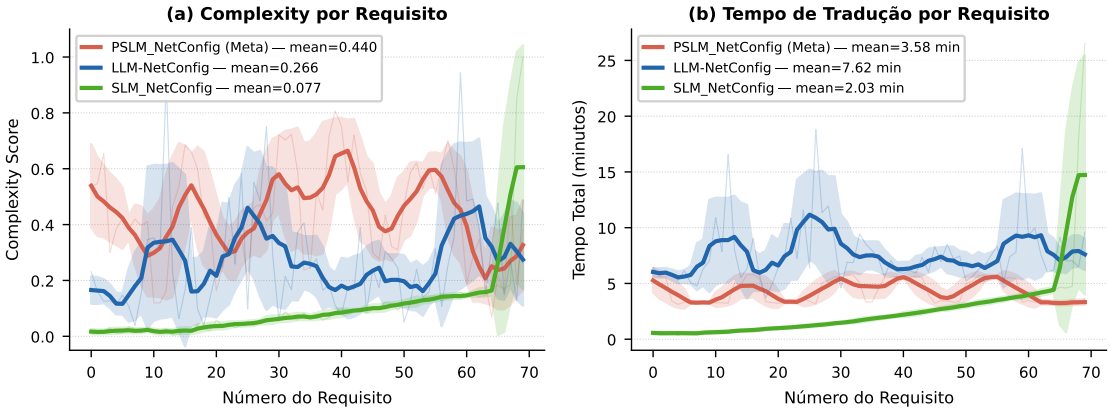

Figura 3. Pontuação de complexidade e tempo de tradução por requisito para os modelos avaliados. A métrica de complexidade combina comprimento e tempo de geração, enquanto o tempo de tradução evidencia que o PSLM_NetConfig mantém desempenho competitivo, com menor variabilidade que o LLM_NetConfig e comportamento temporal comparável ao SLM_NetConfig.

apresenta clara vantagem em relação ao **LLM-NetConfig**, cuja curva exibe maior variabilidade e picos de latência mais elevados. Em relação ao **SLM_NetConfig**, observa-se um comportamento comparável, com diferenças entre requisitos específicos, o que sugere proximidade de desempenho nessa dimensão.

Esse resultado é relevante porque indica que a combinação de *Prompt Tuning* com prompting estruturado pode alcançar tempos de resposta competitivos mesmo sem recorrer a ajuste fino completo. Assim, a abordagem proposta preserva o benefício de uma adaptação mais leve em termos de treinamento e uso de recursos, ao mesmo tempo em que mantém eficiência temporal compatível com métodos especializados para geração de configurações de rede.

4.5. Custo Computacional de Treinamento

O **PSLM_NetConfig**, baseado em Zephyr-7B com *Prompt Tuning*, apresenta uma relação favorável entre desempenho e custo de adaptação quando comparado aos baselines. Enquanto o **LLM-NetConfig** requer aproximadamente **4h45min** de treinamento com o modelo completo em fp16 e cerca de **14 GB** de memória de GPU, e o **SLM_NetConfig** demanda cerca de **2h30min** com LoRA ($r = 64$), quantização 4-bit NF4 e aproximadamente **8 GB** de memória, o modelo proposto conclui o treinamento em apenas $\sim$**42 minutos**, com **204.800 parâmetros treináveis** e consumo estimado de apenas **4,5 GB** de memória, mantendo os pesos do modelo base completamente congelados. Em termos proporcionais, isso representa uma redução de cerca de **72%** no tempo de treinamento e **44%** no uso de memória em relação ao **SLM_NetConfig**, além de uma redução de aproximadamente **85%** no tempo e **68%** na memória quando comparado ao **LLM-NetConfig**. Esses resultados reforçam que o *Prompt Tuning* constitui uma estratégia de adaptação substancialmente mais leve, capaz de preservar competitividade experimental com menor custo computacional e maior viabilidade prática em cenários com restrição de recursos.

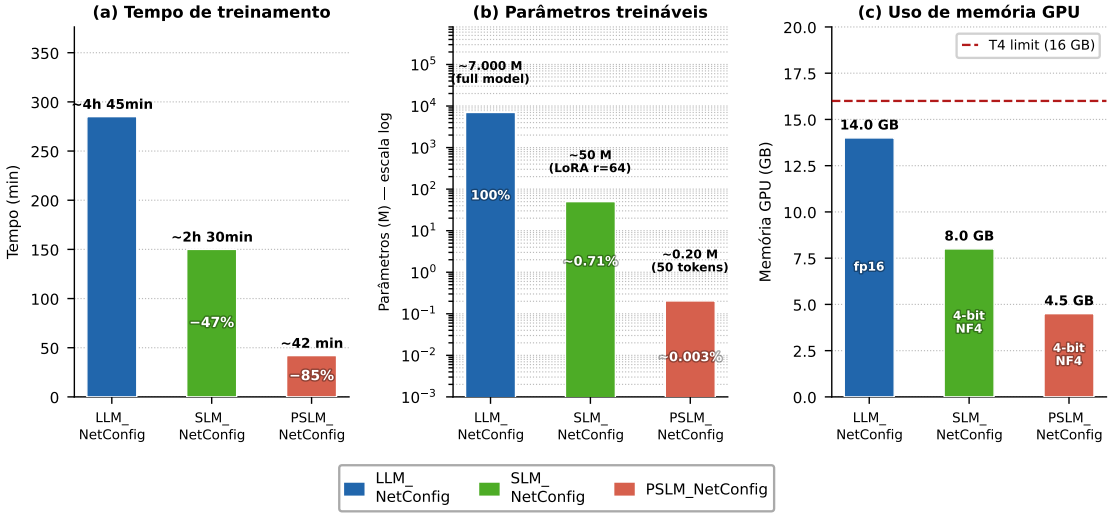

Figura 4. Custo computacional de treinamento dos modelos avaliados: (a) tempo de treinamento, (b) número de parâmetros treináveis em escala logarítmica e (c) uso estimado de memória de GPU. A figura evidencia que o PSLM_NetConfig apresenta uma alternativa mais leve de adaptação em comparação às abordagens baseadas em fine-tuning.

5. Conclusões

Este trabalho apresentou o **PSLM_NetConfig** como uma abordagem baseada em *Prompt Tuning* e prompting estruturado para a geração autônoma de configurações Cisco IOS a partir de requisitos em linguagem natural, sem a necessidade de ajuste completo dos pesos do modelo base Zephyr-7B-β. Os resultados obtidos mostram que, particularmente com o uso de *Meta Prompting*, a proposta alcança desempenho competitivo nas métricas de acurácia sintática e acurácia de objetivo, aproximando-se do **SLM_NetConfig** mesmo com uma estratégia de adaptação substancialmente mais leve. Além disso, o modelo proposto demanda apenas **204.800 parâmetros treináveis**, o que corresponde a aproximadamente **0,003%** dos parâmetros do modelo base, com menor custo de adaptação e menor tempo de treinamento, o que reforça sua viabilidade em cenários com restrições de recursos computacionais.

Em conjunto, esses resultados indicam que a combinação entre *Prompt Tuning* e estruturas explícitas de prompt constitui uma alternativa eficiente para a especialização de modelos no domínio de configuração de redes. Mais do que reduzir o custo computacional, essa abordagem demonstra que é possível obter saídas competitivas explorando o conhecimento já presente no modelo pré-treinado, sem recorrer a processos completos de ajuste fino. Dessa forma, o **PSLM_NetConfig** se mostra uma solução promissora para contextos que exigem readaptações frequentes, implantação local ou maior economia de recursos, preservando um equilíbrio favorável entre desempenho, simplicidade de adaptação e viabilidade prática.

Agradecimentos

Este trabalho foi parcialmente financeado pelo INCT de Redes de Comunicações Inteligentes e Internet das Coisas (ICoNIoT) via CNPq (405940/2022-0) e CAPES

(88887.954253/2024-00); pelo Projeto Temático FAPESP IDRIC (Inteligência Distribuída em Redes de Comunicação e Internet das Coisas), processo 23/00673-7; e pelo PIND da UNICAMP (3406/23).

Referências

Bimo, F. A., Galdon, M. A. C., Lai, C.-K., Cheng, R.-G., and Chong, E. K. (2025). Intent-based network for ran management with large language models. *arXiv preprint arXiv:2507.14230*.

Cisco Systems (2006). Cisco ios (internetwork operating system). `https://www.cisco.com/c/pt_br/support/docs/ios-nx-os-software/ios-software-releases-110/13178-15.html`. Documento ID: 13178. Última actualización: 2 de febrero de 2006.

Cisco Systems (2024). Cisco ios xe 16 release notes. `https://www.cisco.com/c/en/us/support/ios-nx-os-software/ios-xe-16/products-release-notes-list.html`. Accessed: 2026-03-31.

de Lima, M. A. P., da Silva, E. A., and da Silva, A. S. (2024). Um estudo sobre o uso de modelos de linguagem abertos na tarefa de recomendação de próximo item. In *Anais do XXXIX Simpósio Brasileiro de Bancos de Dados (SBBD)*, pages 510–522, Porto Alegre, RS, Brasil. Sociedade Brasileira de Computação (SBC).

Dettmers, T. et al. (2023). Qlora: Efficient finetuning of quantized llms. In *Proceedings of NeurIPS*. arXiv:2305.14314.

El Rajab, M., Yang, L., and Shami, A. (2024). Zero-touch networks: Towards next-generation network automation. *Computer Networks*, 243:110294.

ETSI ISG ZSM (2018). Zero touch network & service management (zsm) standards. Technical report, ETSI.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., et al. (2022). Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3.

Huang, L., Yu, W., Ma, W., Zhong, W., Feng, Z., Wang, H., Chen, Q., Peng, W., Feng, X., Qin, B., and Liu, T. (2025). A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Trans. Inf. Syst.*, 43(2).

Huang, S., Yang, K., Qi, S., and Wang, R. (2024). When large language model meets optimization. *Swarm and Evolutionary Computation*, 90:101663.

Hugging Face H4 Team (2023). Zephyr-7b-beta: A dpo-aligned 7b parameter language model. `https://huggingface.co/HuggingFaceH4/zephyr-7b-beta`. Accessed: 2026-03-31.

Lester, B., Al-Rfou, R., and Constant, N. (2021). The power of scale for parameter-efficient prompt tuning. In *Proceedings of EMNLP*, pages 3045–3059.

Lira, O. G., Caicedo, O. M., and da Fonseca, N. L. (2024). Large language models for zero touch network configuration management. *IEEE Communications Magazine*, 63(7):146–153.

Lira, O. G., Caicedo, O. M., and da Fonseca, N. L. S. (2025). Network self-configuration based on fine-tuned small language models. *IEEE Open Journal of the Communications Society*.

Liyanage, M., Pham, Q.-V., Dev, K., Bhattacharya, S., Maddikunta, P. K. R., Gadekallu, T. R., and Yenduri, G. (2022). A survey on zero touch network and service management (zsm) for 5g and beyond networks. *Journal of Network and Computer Applications*, 203:103362.

Pang, L., Yang, C., Chen, D., Song, Y., and Guizani, M. (2020). A survey on intent-driven networks. *IEEE Access*, 8:22862–22873.

Parthasarathy, V. B., Zafar, A., Khan, A., and Shahid, A. (2024). The ultimate guide to fine-tuning llms from basics to breakthroughs: An exhaustive review of technologies, research, best practices, applied research challenges and opportunities. *arXiv preprint arXiv:2408.13296*.

Suzgun, M. and Kalai, A. T. (2024). Meta-prompting: Enhancing language models with task-agnostic scaffolding.

Tonmoy, S., Zaman, S., Jain, V., Rani, A., Rawte, V., Chadha, A., and Das, A. (2024). A comprehensive survey of hallucination mitigation techniques in large language models. *arXiv preprint arXiv:2401.01313*, 6.

Wang, C., Scazzariello, M., Farshin, A., Kostic, D., and Chiesa, M. (2023). Making network configuration human friendly. *arXiv preprint arXiv:2309.06342*.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E. H., Le, Q., and Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.

XU, L., XIE, H., QIN, S., TAO, X., and WANG, F. (2026). Parameter-efficient fine-tuning methods for pretrained language models : A critical review and assessment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Publisher Copyright: © 1979-2012 IEEE.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. (2023). React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Zhang, Y. (2024). Meta prompting for ai systems. *arXiv preprint arXiv:2311.11482*.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. P., Zhang, H., Gonzalez, J. E., and Stoica, I. (2023). Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*.