Uma avaliação empírica abrangente de LLMs leves para análise de logs de software

Resumo — A análise sintática de logs, uma etapa fundamental na detecção de anomalias em logs de software, desempenha um papel crítico na extração de modelos estruturados a partir de dados de log não estruturados. Embora avanços recentes tenham demonstrado a eficácia de Modelos de Linguagem Ampla (LLMs) em tarefas de análise sintática de logs, a pesquisa atual permanece limitada pelo uso de conjuntos de dados de referência desatualizados e pela falta de investigação sistemática sobre as capacidades de LLMs menores e de código aberto. Este estudo apresenta uma avaliação empírica abrangente de dez analisadores sintáticos leves de código aberto baseados em LLMs, com tamanhos de parâmetros variando de 0,5B a 9B, em seis conjuntos de dados recém-construídos que representam ambientes de log modernos de grande escala e de baixo custo. Nossos resultados revelam uma variabilidade significativa na precisão de análise (PA) e na pontuação F1 da precisão do modelo (FTA), com modelos atingindo médias de até 70,1% PA e 66,8% FTA em conjuntos de dados de referência como o LHSB, mas caindo para 58,7% PA e 53,1% FTA em conjuntos de dados mais diversos como o FALL. Logs mais longos e aqueles contendo variáveis mais dinâmicas também levaram a declínios perceptíveis no desempenho. Notavelmente, ambos os métodos de aprendizado de poucos disparos melhoraram o desempenho em uma média de 7,46% PA. A abordagem de aprendizado de poucos disparos baseada em exemplos superou ligeiramente a contraparte baseada em regras, com a melhoria mais significativa observada no conjunto de dados FALL, onde alcançou um aumento de 14,2% na precisão de análise (PA). Este estudo empírico enfatiza a importância da diversidade do conjunto de dados e do aprendizado de poucos disparos direcionado para aprimorar as capacidades de generalização de LLMs em tarefas de análise de logs do mundo real, oferecendo insights acionáveis para pesquisadores e profissionais em engenharia de software.

Termos de índice — Análise de logs, Modelos de Linguagem Grandes (LLMs), Aprendizado de poucos disparos, extração de modelos, diversidade de conjuntos de dados

I. INTRODUÇÃO

Os logs de software são uma fonte vital de informações para garantir a estabilidade e a confiabilidade dos sistemas modernos, permitindo que os desenvolvedores monitorem o comportamento do sistema, detectem anomalias e diagnostiquem falhas. [1]. Como tal, eles servem como um recurso crítico para obter insights sobre o comportamento do sistema, identificar erros e descobrir gargalos de desempenho [2]. A detecção eficiente de anomalias é importante para identificar falhas potenciais, violações de segurança ou ineficiências operacionais antes que elas se transformem em problemas críticos [3]–[5]. À medida que os sistemas de software se tornam cada vez mais complexos, técnicas eficazes de análise de logs e detecção de anomalias são essenciais para garantir a estabilidade do sistema.

A análise de log é uma etapa crucial na detecção de anomalias, pois a precisão do processo de análise impacta diretamente a eficácia das tarefas subsequentes de detecção de anomalias [6], [7].
Se os logs forem analisados incorretamente, informações valiosas podem ser perdidas, levando a um desempenho ruim dos modelos de detecção de anomalias [8]. Os analisadores sintáticos tradicionais baseados em regras [9]–[12] têm sido amplamente usados para análise de logs, mas pesquisas recentes mostraram que os analisadores sintáticos baseados em Large Language Model (LLM) [13]–[16] geralmente superam as abordagens baseadas em regras em termos de precisão e flexibilidade. Os LLMs podem alavancar grandes quantidades de dados e aprender padrões complexos, o que aumenta sua capacidade de lidar com uma ampla gama de formatos e estruturas de log [16].
Apesar de sua promessa, a aplicação de LLMs para análise de logs enfrenta vários desafios, incluindo dependência de conjuntos de dados desatualizados, altos custos computacionais e exploração limitada de modelos menores e de código aberto.

Limitações dos Conjuntos de Dados de Avaliação Existentes: As avaliações de analisadores baseados em LLM têm se baseado amplamente em conjuntos de dados fornecidos pela plataforma LogHub [17], que são conhecidos por apresentarem diversas restrições [18]. Uma limitação significativa é que esses conjuntos de dados consistem principalmente em registros de sistemas com mais de uma década de existência, apresentando formatos de registro relativamente uniform Consequentemente, o desempenho observado nesses conjuntos de dados pode não ser generalizado adequadamente para sistemas de software modernos, que normalmente envolvem tecnologias mais avançadas e estruturas de log diversas. Além disso, os LLMs testados nesses conjuntos de dados podem apresentar desempenho inflado, pois os modelos podem ter encontrado formatos de log semelhantes durante o pré-treinamento, levando à "memorização" de padrões específicos [13]. Além disso, os conjuntos de dados de referência existentes concentram-se predominantemente em logs de sistemas interconectados de larga escala, como aplicativos de nível empresarial. Embora valiosos para contextos específicos, estudos recentes destacam a necessidade da análise de logs em ambientes menores e com poucos recursos [19]. A detecção eficaz de anomalias é igualmente crítica em configurações de laboratório, implantações de software em estágio inicial, pequenas aplicações industriais e sistemas embarcados, onde menos logs são gerados, mas interrupções operacionais podem ser dispendiosas [20]. Além disso, os logs desses sistemas de menor escala normalmente apresentam maior diversidade, aumentando a complexidade da análise sintática e exigindo maior precisão.

Desafios com os analisadores sintáticos atuais baseados em LLM: Os analisadores sintáticos atuais baseados em LLM dependem fortemente de modelos de larga escala, como o ChatGPT [13], [14], para uma análise sintática de log eficaz. Embora esses modelos tenham demonstrado desempenho de ponta, eles apresentam diversas desvantagens. O desafio mais significativo reside em seu custo e nas demandas computacionais [21]. Esses modelos de larga escala podem ser caros para implementar, especialmente em cenários de tempo real, onde o processamento rápido de logs é necessário. Para organizações com orçamentos limitados ou sistemas de pequena escala, o custo de utilização de modelos de grande porte pode ser proibitivo.
Além disso, a eficiência desses grandes modelos em aplicações em tempo real é uma preocupação, pois seus tempos de inferência podem não atingir a velocidade necessária para tarefas de detecção de anomalias sensíveis ao tempo. Muitos LLMs menores e de código aberto, que exigem menos recursos, não foram explorados completamente para a tarefa de análise de logs.

Estudo Empírico: Para abordar essas limitações, este estudo apresenta uma avaliação empírica abrangente de analisadores de log baseados em LLM de código aberto, com foco em modelos leves. Para avaliar a eficácia desses modelos em uma variedade de ambientes, construímos seis conjuntos de dados diversos que refletem uma gama de fontes de dados de log. O primeiro conjunto de dados é baseado no benchmark LogHub, fornecendo uma referência bem estabelecida para avaliar o desempenho do analisador. O segundo e o terceiro conjuntos de dados consistem em logs de sistemas interconectados modernos, permitindo uma avaliação de quão bem os modelos podem lidar com ambientes de software contemporâneos mais complexos. Os três conjuntos de dados finais representam cenários de poucos recursos, incluindo logs de configurações de laboratório, lançamentos de software em estágio inicial e sistemas embarcados. Esses conjuntos de dados são projetados para refletir melhor os desafios enfrentados por sistemas menores, onde o volume de logs é limitado e alta precisão de análise é necessária. Em termos de modelos, selecionamos dez LLMs leves de código aberto, incluindo Deepseek-R1-1.5B, Gemma-2-2B, Gemma-2-9B, Llama-3.2-1B, Llama-3.2-3B, Llama-3.1-8B, Qwen-2.5-0.5B, Qwen-2.5-1.5B, Qwen-2.5-3B e Qwen-2.5-7B. Esses modelos representam um amplo espectro de opções de código aberto, oferecendo insights sobre o desempenho de LLMs de médio porte em tarefas de análise sintática de logs. Por meio dessa avaliação abrangente, pretendemos determinar o desempenho desses modelos em diferentes conjuntos de dados e ambientes, com foco específico em como o aprendizado de poucas tentativas pode melhorar sua precisão em tarefas de análise sintática de logs.

Principais descobertas: Com base em nossos resultados, apresentamos as seguintes descobertas principais: (1) Sensibilidade do conjunto de dados: O desempenho dos analisadores sintáticos baseados em LLM de código aberto varia significativamente entre os conjuntos de dados, com a maior precisão observada em conjuntos de dados de referência, como o LHSB, enquanto o desempenho diminui em conjuntos de dados mais diversos e complexos, como o FALL. (2) Impacto das características do log: O desempenho da análise é substancialmente influenciado pelas características do log. Em particular, entradas de log mais longas e aquelas com maior frequência de variáveis dinâmicas reduzem consistentemente a precisão do modelo. (3) Eficácia do aprendizado de poucos disparos: Ambos os métodos de aprendizado de poucos disparos melhoram consistentemente a precisão da análise de todos os analisadores sintáticos baseados em LLM testados, indicando que a adaptação do modelo melhora significativamente o desempenho em comparação com cenários de zero disparos. (4) Comparação dos métodos de poucos disparos: Entre as duas estratégias de aprendizado de poucos disparos, o método baseado em exemplos supera ligeiramente o método baseado em regras, especialmente em conjuntos de dados com maior complexidade e diversidade, como o FALL. No geral, nossas principais contribuições podem ser resumidas em quatro pontos principais.

- Realizamos um estudo empírico abrangente para avaliar o desempenho de analisadores sintáticos (LLMs) leves de código aberto com parâmetros menores para análise de logs de software em vários conjuntos de dados e ambientes. • Construímos seis novos conjuntos de dados, incluindo dados de referência e dados de log de sistemas modernos e de baixo recurso, para fornecer uma avaliação diversificada para analisadores sintáticos. • Somos os primeiros a avaliar de forma abrangente o desempenho de LLMs leves de código aberto para análise de logs.
preenchendo uma lacuna na pesquisa existente.
- Os experimentos extensivos demonstram que analisadores leves baseados em LLM podem atingir análise de log de alta precisão e exibir melhorias adicionais com dois métodos de aprendizado de poucas tentativas.

II. CONTEXTO

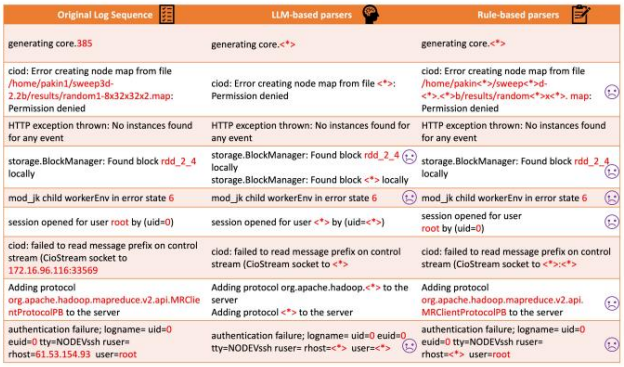
A. Coleta de Logs A

natureza heterogênea das estruturas de logs, decorrente de diferentes práticas e padrões de registro adotados por diferentes desenvolvedores, introduz considerável variabilidade entre os logs coletados [22]. Essa heterogeneidade representa desafios significativos para o desenvolvimento de modelos de detecção de anomalias capazes de desempenho consistente e preciso em diversos ambientes de registro. Embora plataformas como o LogHub [17] tenham fornecido diversos conjuntos de dados de código aberto, esses conjuntos de dados sofrem de diversas limitações, notadamente sua natureza desatualizada e foco predominante em logs gerados por sistemas interconectados de larga escala.
Como resultado, eles podem não conseguir capturar as características dos sistemas de software contemporâneos ou abordar cenários que envolvam ambientes de pequena escala ou com recursos limitados [19].
Em particular, conjuntos de dados derivados de sistemas interconectados modernos são essenciais, visto que esses sistemas frequentemente adotam arquiteturas avançadas e produzem registros com padrões complexos e dinâmicos, o que representa novos desafios para análise sintática e detecção de anomalias. Além disso, conjuntos de dados que refletem cenários com poucos recursos — como ambientes de laboratório, implantações de software em estágio inicial e sistemas embarcados — são igualmente importantes. Esses ambientes normalmente geram menos logs, mas apresentam maior variabilidade. Devido às atualizações frequentes de software e aos fluxos de trabalho de produção complexos, a análise precisa de logs torna-se ainda mais crítica nesses contextos.

B. Análise de Log

A análise sintática de logs refere-se à transformação automática de mensagens de log brutas em representações estruturadas, identificando componentes constantes (modelos de eventos) e componentes variáveis (parâmetros) [23]. Pesquisas indicaram que mais de 30% das mensagens de log individuais contêm mais de três parâmetros variáveis, com aproximadamente 10% excedendo até cinco variáveis [24]. Essa variabilidade complica a análise sintática precisa, pois distinguir com precisão cada parâmetro dentro dessas entradas de log estruturadas dinamicamente está longe de ser trivial [3]. Estudos anteriores [7], [8] mostraram que a qualidade da análise sintática de logs influencia significativamente a eficácia da detecção de anomalias e outras tarefas de análise de logs posteriores [2], [16]. Como tal, a análise sintática de logs é amplamente considerada um componente fundamental do pipeline de análise de logs. Uma variedade de abordagens de análise sintática de logs foi proposta na literatura, amplamente categorizadas em dois grupos: métodos baseados em regras (por exemplo, Drain [9]) e métodos baseados em LLM. Conforme mostrado na Figura 1, embora os métodos baseados em regras sejam eficazes em muitos ambientes de registro, eles não têm a flexibilidade necessária para lidar com formatos de registro diversos ou mais complexos.

Recentemente, o surgimento de Large Language Models (LLMs), exemplificados pelo ChatGPT [25], ofereceu perspectivas promissoras.



Original Log Sequence	LLM-based parsers	Rule-based parsers
generating core.385	generating core.<*>	generating core.<*>
ciod: Error creating node map from file /home/pakin1/sweep3d-2.2b/results/random1-8x32x32x2.map: Permission denied	ciod: Error creating node map from file <*>: Permission denied	ciod: Error creating node map from file /home/pakin<*>/sweep<*>d-<*>.<*>b/results/random<*>x<*>. map: Permission denied
HTTP exception thrown: No instances found for any event	HTTP exception thrown: No instances found for any event	HTTP exception thrown: No instances found for any event
storage.BlockManager: Found block rdd_2_4 locally	storage.BlockManager: Found block rdd_2_4 locally storage.BlockManager: Found block <*> locally	storage.BlockManager: Found block rdd_2_4 locally
mod_jk child workerEnv in error state 6	mod_jk child workerEnv in error state 6	mod_jk child workerEnv in error state 6
session opened for user root by (uid=0)	session opened for user <*> by (uid=<*>)	session opened for user root by (uid=0)
ciod: failed to read message prefix on control stream (CioStream socket to 172.16.96.116:33569	ciod: failed to read message prefix on control stream (CioStream socket to <*>	ciod: failed to read message prefix on control stream (CioStream socket to <*>:<*>
Adding protocol org.apache.hadoop.mapreduce.v2.api.MRClientProtocolPB to the server	Adding protocol org.apache.hadoop.<*> to the server Adding protocol <*> to the server	Adding protocol org.apache.hadoop.mapreduce.v2.api.MRClientProtocolPB to the server
authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=61.53.154.93 user=root	authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=<*> user=<*>	authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=<*> user=root

Fig. 1: Desempenho dos dois tipos de analisadores mais utilizados. Variáveis dinâmicas são destacadas em **vermelho.**

alternativas às abordagens baseadas em regras [26], [27]. LogGPT [28] apresenta uma nova abordagem para detecção de anomalias baseada em log, aproveitando os recursos de interpretação de linguagem do ChatGPT-3.5. Os resultados mostram desempenho promissor e forte interpretabilidade, fornecendo insights valiosos sobre o potencial de modelos baseados em prompts para análise de dados de log. No entanto, o desempenho do Log-GPT no conjunto de dados BGL é abaixo do ideal, ressaltando a necessidade de melhorias adicionais nessa abordagem.

LogPPT [14] apresenta uma abordagem de aprendizado de poucos passos baseada em prompts para análise sintática de logs. Este método captura modelos e parâmetros de log usando dados minimamente rotulados, superando as técnicas tradicionais em eficácia e eficiência em diversos conjuntos de dados de logs públicos. DivLog [24] utiliza aprendizado em contexto com LLMs para aprimorar a análise sintática de logs, aproveitando amostras pequenas e diversas e a construção de prompts sem treinamento, alcançando desempenho de ponta em precisão e exatidão de modelos em múltiplos conjuntos de dados de grande escala.

Além disso, LILAC [13] integra LLMs com um cache de análise adaptável e ICL, melhorando significativamente a precisão do modelo, a eficiência da análise e reduzindo a dependência de consultas LLM.

Comparados aos analisadores sintáticos baseados em regras mostrados na Figura 1, os analisadores sintáticos baseados em LLM potencializam sua capacidade de reconhecer padrões textuais complexos e variados, oferecendo vantagens significativas em relação aos métodos tradicionais, especialmente ao analisar logs com alta variabilidade. No entanto, a implantação prática de modelos tão grandes é limitada por custos computacionais substanciais e requisitos de recursos, limitando sua aplicabilidade em cenários de tempo real ou com poucos recursos. Além disso, a pesquisa existente tem se concentrado principalmente em LLMs de grandes parâmetros, enquanto o potencial de LLMs leves de código aberto permanece amplamente inexplorado. Essa lacuna de pesquisa apresenta uma oportunidade para investigar se LLMs de menor escala, com sua maior viabilidade de implantação, podem atingir desempenho de análise sintática comparável ou superior por meio de aprendizado direcionado de poucas tentativas.

III. DESENHO DO ESTUDO

A. Cenários e Construção de Conjuntos de Dados

Construímos seis conjuntos de dados diversos para avaliar de forma abrangente o desempenho de modelos de análise de log em diferentes ambientes de registro. Os conjuntos de dados incluem: (1) o conjunto de dados **LogHub Subset Benchmark (LHSB) ,** derivado de benchmarks estabelecidos na plataforma LogHub; (2) o conjunto de dados **Distributed System Logs (DSL) ,** coletado de um ambiente de sistema distribuído; (3) o conjunto de dados **Large-Scale Web Application Logs (LSWAL) ,** representando logs de um sistema interconectado moderno; (4) o **Educational App Interactive Log Dataset (EAILD),** coletado de uma implantação de software educacional; (5) o **Intelligent Laboratory Management System Dataset (ILMSD),** coletado de um sistema de gerenciamento de laboratório de pesquisa; e (6) o conjunto de dados **Factory Assembly Line Logs (FALL) ,** originário de um ambiente de manufatura industrial. A Tabela I resume os principais detalhes de cada conjunto de dados construído.

1) Conjunto de dados LHSB: Para o conjunto de dados LHSB, nosso objetivo era construir um corpus de logs brutos que refletisse com precisão a complexidade estrutural e a distribuição de eventos características das operações de sistemas no mundo real. Embora a plataforma LogHub forneça 2.000 entradas de log manualmente selecionadas por conjunto de dados, esses subconjuntos frequentemente excluem anomalias raras, particularmente aquelas que ocorrem em intervalos curtos ou sob condições limitadas.

Para superar essa limitação, selecionamos cinco conjuntos de dados LogHub [17] — HDFS [29], Hadoop [30], OpenStack [31], BGL [32] e Thunderbird [32] — todos os quais incluem rastros de log completos acompanhados por rótulos de anomalias de verdade fundamental. Nosso processo de construção de dados seguiu uma estratégia de duas etapas: (1) extração de anomalias baseada em intervalos e (2) integração de conjuntos de dados com uma proporção de anomalias controlada.

Na primeira etapa, extraímos todas as entradas de log que ocorreram dentro dos intervalos anômalos rotulados. Para cada conjunto de dados, os arquivos de metadados associados foram analisados para determinar os carimbos de data/hora de início e término de cada janela anormal. Todas as linhas de log dentro desses intervalos foram mantidas em seu formato original, garantindo uma cobertura abrangente de anomalias frequentes e raras que podem ter sido omitidas nos subconjuntos de benchmark padrão.

Na segunda etapa, integramos os registros de anomalias extraídos com as 32.000 entradas de registro provenientes de 16 subconjuntos de benchmark do LogHub. Para aproximar um desequilíbrio de classe realista, removemos um grande número de registros normais semelhantes e duplicamos seletivamente uma parte dos registros de anomalias antes de mesclá-los no conjunto de dados combinado. Isso resultou em uma proporção aproximada de 9:1 de entradas normais para anômalas, refletindo a distribuição assimétrica comumente observada em ambientes de produção.

Excluímos os conjuntos de dados restantes do LogHub do nosso pipeline de processamento devido à ausência de anotações de anomalias confiáveis. Como resultado, nosso conjunto de dados final compreende 9.047 entradas de log brutas, que foram selecionadas sem aplicar nenhuma análise sintática baseada em modelo ou filtragem baseada em rótulo, além da identificação inicial dos intervalos de log.

2) Conjuntos de dados DSL e LSWAL: Os conjuntos de dados DSL e LSWAL foram coletados de sistemas contemporâneos de larga escala para capturar a variabilidade de logs recentes. O DSL inclui logs de uma arquitetura moderna de microsserviços distribuídos, enquanto o LSWAL abrange logs de um importante sistema de aplicação web interconectado, ambos coletados ao longo de um período contínuo em agosto de 2024. Para os conjuntos de dados DSL e LSWAL, que continham cada um mais de um milhão de entradas de log brutas, enfrentamos dois desafios principais:

TABELA I: Resumo dos conjuntos de dados construídos.

Conjunto de dados	Abreviação	Categoria	Anomalias de Quantidade		Média de comprimento (caracteres)
Referência de subconjunto do LogHub	**LHSB**	Benchmark do LogHub	9.047	961 (10,6%)	173,8
Logs do Sistema Distribuído	**DSL**	Sistema moderno de grande escala	7.993	1.149 (14,4%) 813	168,5
Logs de aplicativos da Web em larga escala	**LSWAL**	Sistema moderno de grande escala	6.039	(13,5%) 938	185,2
Conjunto de dados de registro interativo do aplicativo educacional	**EAILD**	Aplicação de poucos recursos	6.285	(14,9% 894	180,5
Conjunto de dados do Sistema de Gestão de Laboratório Inteligente	**ILMSD**	Sistema de baixo recurso	7.480	(12,0%) 917	137,4
Registros da linha de montagem da fábrica	**CAIR**	Sistema de baixo recurso	5.319	(17,2%)	225,8

o enorme volume de dados e a alta redundância causada por numerosos logaritmos normais repetidos ou muito semelhantes mensagens. Para construir conjuntos de dados menores e de alta qualidade enquanto preservando a diversidade, empregamos um método de amostragem escalável com base em Processos de Pontos Determinantes (DPP) [33].

No início, utilizamos o modelo Sentence-BERT [34] (especificamente all-MiniLM-L6-v2) para codificar cada entrada de log em um vetor semântico de tamanho fixo. Para abordar o problema computacional sobrecarga de operação em incorporações de grande escala, nós Análise de Componentes Principais (ACP) aplicada [35] para reduzir a dimensionalidade vetorial de 384 para 50, mantendo a maior parte características informativas. Em seguida, calculamos uma matriz de kernel de similaridade usando similaridade de cosseno [36], que capturou a semântica relações entre entradas de log e estabeleceu a base para seleção de diversidade mais eficiente e significativa por meio do DPP amostragem.

Para construir um subconjunto diverso e representativo, empregamos uma aproximação de baixo nível do algoritmo de amostragem DPP, que prioriza a seleção de semanticamente distintos mensagens de log. Este método reduziu efetivamente a redundância ao mesmo tempo em que garante ampla cobertura em vários formatos de log e tipos de anomalias. Através deste processo, selecionamos modelos refinados conjuntos de dados nos quais aproximadamente 90% das entradas de log representam usuários típicos e comportamento do sistema, enquanto o restante 10% capturam padrões mais incomuns ou diversos. O resultado os conjuntos de dados contêm 7.993 logs para DSL e 6.039 logs para LSWAL, encontrando um equilíbrio entre eficiência, diversidade e representatividade.

3) Conjuntos de dados de cenários de poucos recursos: os três restantes conjuntos de dados (EAILD, ILMSD, FALL) representam cenários de poucos recursos onde os registros são limitados em volume, mas diversos em estrutura.

O conjunto de dados **EAILD** foi coletado de um recém-lançado aplicativo educacional em junho de 2024. O aplicativo oferece suporte a recursos como como aulas interativas, questionários e acompanhamento de progresso, e é implantado em vários dispositivos móveis e redes. um período de depuração focado de três dias, 73.626 mensagens de log foram capturados, abrangendo interações do usuário, status do serviço de back-end e eventos de erro. Uma parcela significativa destes os registros consistiam em entradas quase idênticas que diferiam apenas em valores de parâmetros (por exemplo, carimbos de data/hora ou IDs de sessão). Para reduzir redundância e aumentar a utilidade avaliativa do conjunto de dados, aplicamos filtragem baseada em frequência para remover modelos de log excessivamente repetidos, mantendo sua variabilidade estrutural. Isso resultou em um conjunto de dados limpo de 6.285 logs, preservando diversidade significativa sem sobrecarregar a avaliação com variações triviais.

O conjunto de dados **ILMSD** foi coletado de uma plataforma de gerenciamento de laboratório usada em pesquisas químicas e de materiais instalações. O sistema, desenvolvido em Java e instrumentado com Apache Log4j, registra o comportamento do sistema de forma estruturada Formato JSON para facilitar a automação e a clareza. Os logs incluem ações do usuário, comandos de controle do instrumento, agendamento de tarefas, e processos de sincronização de dados. Mais de 50 horas de operação período, foram registradas 54.637 entradas. Aplicando o mesmo estratégia de filtragem com reconhecimento de frequência, como com EAILD, reduzimos entradas redundantes e construiu um conjunto de dados final de 7.480 logs, manter uma amostra representativa de diversos tipos de log dentro um ambiente controlado.

O conjunto de dados **FALL** foi obtido de uma fábrica operacional sistema de monitoramento de linha de montagem. Este ambiente integra vários subsistemas de hardware e software responsáveis por tarefas como corte, montagem e verificação de qualidade de componentes industriais. As mensagens de log deste sistema são altamente heterogêneo, refletindo o status do hardware em tempo real, comandos de controle e leituras ambientais. Devido à natureza crítica desses processos, os logs são altamente contextuais e diverso, com muito poucos casos de estrutura repetida. Nós coletou todos os registros gerados em três dias consecutivos de produção em outubro de 2024, totalizando 5.319 entradas. Ao contrário do outros conjuntos de dados, nenhuma filtragem foi necessária devido à natureza variabilidade dos troncos, tornando o FALL o mais estruturalmente conjunto de dados diversificado e realista em nossa estrutura de avaliação.

B. Projeto de analisadores baseados em LLM

LLMs leves de código aberto com parâmetros menores têm mostrou grande potencial para aplicações de análise de log devido ao seu menores requisitos computacionais, custos de implantação reduzidos, e maior flexibilidade para aprendizagem de poucos disparos [37]. Além disso, Eles podem atender aos requisitos de velocidade necessários para tarefas de detecção de anomalias sensíveis ao tempo. Para explorar sua eficácia, avaliamos dez LLMs representativos de código aberto em tarefas de análise de log de software.

Para avaliar o desempenho de LLMs de código aberto [38]–[41] para análise de log, selecionamos dez modelos de diferentes tamanhos de parâmetros, que incluem **Deepseek-R1-1.5B, Gemma-2-2B, Gemma-2-9B, Llama-3.2-1B, Llama-3.2-3B, Llama-3.1-8B, Qwen-2.5-0.5B, Qwen-2.5-1.5B, Qwen-2.5-3B** e **Qwen-2.5-7B.** Conforme mostrado na Tabela II, esses modelos representam uma variedade de tamanhos de parâmetros pequenos, permitindo uma avaliação abrangente em várias restrições computacionais e de memória. Aproveitando arquiteturas de ponta, esses modelos demonstram forte potencial para aplicações de análise de log, particularmente à medida que a complexidade e a diversidade dos logs de software aumentam. Em tais contextos, onde a geração precisa de modelos a partir de mensagens de log ricas em variáveis são cruciais, esses modelos são adequados para lidar com os desafios em evolução. Em detalhes, Deepseek-R1-1.5B [38], que é otimizado para inferência eficiente em

tarefas gerais de processamento de texto; Gemma-2-2B e Gemma-2-9B [39], desenvolvido pelo Google DeepMind, enfatizando a escalabilidade e robustez para ambientes computacionais menores; e Lhama-3.2-1B, Lhama-3.2-3B e Lhama-3.1-8B [40] de Meta, conhecida por equilibrar desempenho e eficiência de recursos. Além disso, avaliamos quatro modelos do Qwen família [41]—Qwen-2,5-0,5B, Qwen-2,5-1,5B, Qwen-2,5-3B, e Qwen-2.5-7B — criado pela Alibaba Cloud, que são projetado para fornecer um desempenho forte em multilíngues e tarefas específicas de domínio, mantendo, ao mesmo tempo, um desempenho computacional modesto demandas. A avaliação desses diversos modelos permite uma análise abrangente do potencial e das limitações de LLMs de código aberto de médio porte para aplicações de análise de logs.

TABELA II: Visão geral do LLM leve de código aberto baseado em Analisadores.

Modelo	Tamanho	Base de Treinamento (Tokens)	Entrada/Saída (Fichas)
Deepseek-R1-1.5B	1.5B	14,8 trilhões	2 64 mil / 8.192
Gemma-2-2B	2B	**Gemma-2-9B**	trilhões 8 8.192 / 8.192
9B	**Llama-3.2-1B**	1B	trilhões 9 8.192 / 8.192
Llama-3.2-3B	3B	**Llama-3.1-8B**	trilhões 9 128 mil / 2.048
8B	**Qwen-2.5-0.5B**	0.5B	trilhões 15 128 mil / 2.048
Qwen-2.5-1.5B	1.5B		trilhões 18 128 mil / 2.048
Qwen-2.5-3B	3B	**Qwen-2.5-7B**	trilhões 18 32 mil / 8.192
7B			trilhões 18 32 mil / 8.192
			trilhões 18 32 mil / 8.192
			trilhões 128 mil / 8.192

Para nossa tarefa de análise de log, projetamos **prompts de tiro zero** para avaliar a capacidade dos modelos de lidar com a análise de log sem qualquer treinamento prévio na tarefa específica. O prompt de tiro zero foi projetado da seguinte forma:

> Você precisa analisar o conteúdo dos logs e substituí-los as variáveis dinâmicas com modelos. Produza os resultados diretamente, sem maiores explicações. Por favor, analise o log modelo da seguinte mensagem de log: [LOG]

Onde [LOG] denota os dados de log brutos a serem analisados, isto o prompt direciona o modelo para identificar componentes dinâmicos dentro do log e convertê-los em modelos estruturados. Ao utilizar o prompt de tiro zero, podemos avaliar as capacidades inerentes dos modelos para executar a tarefa sem a necessidade de exemplos adicionais. Esta abordagem serve como um linha de base para comparar o desempenho bruto de cada modelo.

C. Aprendizagem de poucos tiros

Nos experimentos seguintes, avaliamos o impacto de usando Aprendizagem em Contexto (ICL) [42] como uma aprendizagem de poucos passos estratégia sobre o desempenho de dez LLMs de código aberto em análise de log. Embora estudos anteriores tenham mostrado que o ICL melhora o desempenho de análise de log de modelos grandes como ChatGPT [43], houve exploração limitada do seu impacto nos LLMs de código aberto que selecionamos. Portanto, nosso objetivo é para investigar mais a fundo se o ICL melhora esses modelos precisão na análise de logs aproveitando exemplos contextuais para orientar seu processo de tomada de decisão. A solicitação do ICL depende ao condicionar o modelo com um pequeno conjunto de exemplos, que demonstrou melhorar o desempenho em tarefas que exigem raciocínio em várias etapas. No contexto de log análise, os prompts ICL auxiliam o modelo no processamento de entradas de log fornecendo exemplos, permitindo-lhe assim identificar e extrair componentes como registros de data e hora, endereços IP, usuários ações e códigos de erro. Este processo de raciocínio estruturado é crucial para identificar com precisão variáveis dinâmicas em logs e substituí-los por modelos apropriados.

Para nossa tarefa específica, os prompts ICL ajudam os modelos examinar cuidadosamente as diferentes partes de uma mensagem de log e compreender as relações estruturais entre eles. Por fornecendo ao modelo um conjunto inicial de exemplos de poucos disparos, orientamos o modelo na aprendizagem do padrão correto de como os logs deve ser analisado. Essa abordagem permite que o modelo adapte seu comportamento de análise para novos logs com estruturas semelhantes, melhorando sua capacidade de gerar modelos de log precisos mesmo quando confrontados com dados invisíveis. Projetamos dois tipos de prompts Few-Shot para este propósito:

1) Prompt de poucos disparos baseado em exemplos: neste método, nós forneceu cinco pares de registros originais e seus correspondentes modelos de log analisados para o modelo. Para garantir uma gama diversificada de exemplos, selecionamos 1.500 entradas de log de todos os conjuntos de dados usando o método DPP. O DPP é uma técnica de seleção que visa maximizar a diversidade do conjunto de dados escolhido selecionando logs que são menos semelhantes entre si [44]. Isso garante que o o modelo é exposto a uma ampla variedade de formatos e padrões de log, evitando que ele se ajuste excessivamente a uma estrutura de log específica. processo envolve o cálculo de similaridades em pares entre todos logs no conjunto de dados (usando uma medida de similaridade como cosseno similaridade ou similaridade de Jaccard) e, em seguida, selecionando logs que minimizar a similaridade geral. Esta abordagem seleciona iterativamente os mais diversos logs, garantindo que o modelo encontre variados padrões.

Após construir um conjunto de dados diversificado com DPP, aplicamos o algoritmo k-Nearest Neighbors (kNN) [45] para identificar o cinco logs mais semelhantes a uma nova entrada de log. Neste método, para a cada novo log, o algoritmo compara o novo log com o registros históricos selecionados e selecionados as k entradas mais semelhantes com base em uma pontuação de similaridade. Ao usar o kNN, garantimos que o modelo se adapta a novos padrões de log e refina sua análise habilidade baseada em exemplos passados. O prompt apresentado ao modelos durante esta fase é o seguinte:

> Você precisa analisar o conteúdo dos logs e substituí-los as variáveis dinâmicas com modelos. Produza os resultados diretamente sem maiores explicações. Aqui estão alguns exemplos:
> Registro original: "[Exemplos de registro original 1, 2, 3, ...]"
> Log analisado: "[Exemplos de log analisado 1, 2, 3, ...]"
> Analise o modelo de log da seguinte mensagem de log:
> [REGISTRO]

Onde [Exemplos de log original] denota uma mensagem de log real que contém variáveis dinâmicas, como registros de data e hora, IP endereços e IDs de usuário, e [Exemplo de log analisado] denota o log de saída com essas variáveis substituídas por estruturadas modelos como "<*>".

2) Prompt de poucos tiros baseado em regras de substituição: neste método, fornecemos um conjunto de regras de substituição explícitas que o o modelo deve seguir para garantir uma análise consistente e precisa

de logs. Essas regras ajudam a padronizar como variáveis dinâmicas específicas devem ser substituídas por modelos. As regras de substituição predefinidas orientam o modelo a reconhecer e substituir componentes como carimbos de data/hora, endereços IP e IDs de usuário, comumente encontrados em dados de log. Por exemplo:

> Substituir todos os registros de data e hora (por exemplo, [Exemplos]) por "HORA"; Substituir todos os endereços IP (por exemplo, [Exemplos]) por "IP"; Substituir IDs de usuário (por exemplo, [Exemplos]) por "USUÁRIO".

Onde [Exemplos] denota exemplos específicos de variáveis dinâmicas, selecionamos os tipos de variáveis dinâmicas presentes em diferentes conjuntos de dados, como registros de data e hora, endereços IP e IDs de usuário. Essas regras servem como instruções claras, reduzindo a ambiguidade no processo de análise de logs e garantindo que o modelo analise os logs de forma estruturada e consistente, independentemente do formato específico do log. Ao oferecer essas regras predefinidas, orientamos as decisões de análise do modelo, permitindo que ele se concentre em elementos específicos que exigem substituição, ignorando componentes irrelevantes. O prompt apresentado aos modelos, incorporando as regras de substituição, é o seguinte:

> Você precisa analisar o conteúdo dos logs e substituir as variáveis dinâmicas por modelos. Exiba os resultados diretamente, sem maiores explicações.
> Consulte estas regras de substituição: Substitua todos os registros de data e hora (por exemplo, [Exemplos 1, 2, 3, ...]) por "HORA"; Substitua todos os endereços IP (por exemplo, [Exemplos 1, 2, 3, ...]) por "IP"; Substitua os IDs de usuário (por exemplo, [Exemplos 1, 2, 3, ...]) por "USUÁRIO".
> .....
> Analise o modelo de log da seguinte mensagem de log:
> [REGISTRO]

D. Design Experimental e Questões de Pesquisa

Nesta seção, apresentaremos as questões de pesquisa elaboradas e a configuração experimental para cada questão de pesquisa.

RQ1: Como os dez analisadores LLM de código aberto executar em seis conjuntos de dados?

Neste estudo, pretendemos avaliar o desempenho de dez analisadores LLM leves de código aberto em seis conjuntos de dados distintos, representando uma gama diversificada de dados de log.
Isso nos permite comparar o desempenho dos modelos tanto em registros de grande volume e larga escala quanto em entradas de registro menores e mais variadas. Aplicamos cada um dos dez modelos — Deepseek-R1-1.5B, Gemma-2-2B, Gemma-2-9B, Llama-3.2-1B, Llama-3.2-3B, Llama-3.1-8B, Qwen-2.5-0.5B, Qwen-2.5-1.5B, Qwen-2.5-3B e Qwen-2.5-7B — para analisar registros dos seis conjuntos de dados a seguir: LHSB, DSL, LSWAL, EAILD, ILMSD e FALL. Além disso, adotamos o GPT-3.5-Turbo, fornecido pela OpenAI, como um LLM de parâmetros grandes de referência para análise de registros.
Avaliamos a precisão da análise medindo a capacidade dos modelos de substituir variáveis dinâmicas pelos modelos corretos em vários formatos de log.

RQ2: Quais fatores de entradas de log influenciam o desempenho da análise de log?

Para explorar quais fatores das entradas de log impactam o desempenho da análise, examinamos uma série de características do log, incluindo o comprimento das entradas de log e a frequência das variáveis dinâmicas. Para o comprimento das entradas de log, categorizamos os logs dos seis conjuntos de dados em três grupos com base na contagem de caracteres: (1) registros com comprimento de 100 caracteres ou menos, (2) registros com comprimentos variando de 100 a 200 caracteres e (3) registros com mais de 200 caracteres. Essa classificação nos permite criar um novo conjunto de dados de avaliação com base no comprimento do registro, o que nos permite determinar como o comprimento dos registros influencia a precisão da análise.

Além disso, alguns logs, apesar de longos, podem não apresentar desafios significativos para os modelos se contiverem poucas variáveis dinâmicas. Para levar isso em conta, introduzimos outro fator: a frequência das variáveis dinâmicas. Dividimos os logs em três categorias com base no número de variáveis dinâmicas que contêm: (1) 1 a 2 variáveis, (2) 3 a 4 variáveis e (3) 5 ou mais variáveis. Essa categorização nos permite criar um segundo conjunto de dados de avaliação que reflete como o número de variáveis dinâmicas influencia o desempenho da análise sintática dos logs.

Em seguida, avaliamos como cada um desses fatores — comprimento do log e frequência de variáveis dinâmicas — afeta a precisão da análise sintática dos modelos em diferentes conjuntos de dados. Para evitar a repetição excessiva de logs semelhantes (ou seja, logs com apenas variáveis dinâmicas diferentes), o que poderia afetar as métricas de avaliação, removemos uma parte dos logs com formatos idênticos, garantindo que não restem mais do que cinco instâncias de cada tipo de log. Ao analisar esses fatores, buscamos identificar quais características do log têm o maior impacto no desempenho da análise sintática, oferecendo insights sobre como o desempenho dos analisadores baseados em LLM varia com diferentes estruturas de log.

RQ3: Até que ponto o aprendizado de poucos disparos melhora o desempenho da análise de log?

Neste estudo, exploramos o impacto do aprendizado de poucas tentativas no desempenho de analisadores sintáticos baseados em LLM para tarefas de análise de logaritmos. Para avaliar isso, ajustamos os modelos selecionados usando os métodos descritos na Seção III-C, onde os modelos são treinados para melhorar seu desempenho na análise de logaritmos. Comparamos o desempenho de dois tipos de modelos ajustados — prompts de poucas tentativas (baseados em exemplos e baseados em regras de substituição) — com seu desempenho de zero tentativas nos mesmos seis conjuntos de dados. Espera-se que o aprendizado de poucas tentativas melhore a precisão da análise, permitindo que os modelos se adaptem mais especificamente aos tipos de logaritmos e às características únicas dos conjuntos de dados, permitindo-nos quantificar os benefícios da adaptação de modelos na análise de logaritmos.

E. Métricas de Avaliação

Para avaliar o desempenho dos dez analisadores LLM de código aberto na análise de log, seguimos estudos recentes [13], [16] e usamos PA e FTA como métricas de avaliação.

1) Precisão de Análise Sintética (PA): A PA mede a capacidade do analisador de extrair modelos com precisão das entradas de log, o que é essencial para tarefas posteriores, como detecção de anomalias e análise de logs. A PA é definida como a proporção de mensagens de log analisadas corretamente em relação ao número total de mensagens de log. Uma mensagem de log é considerada analisada corretamente se, e somente se, todos os tokens dos modelos e variáveis forem identificados corretamente. PA = onde um "log $\frac{\text{Número de logs analisados corretamente}}{\text{Número total de logs}}$ analisado corretamente" significa que todas as variáveis dinâmicas são identificadas e substituídas com precisão.

2) Pontuação F1 de Precisão do Modelo (FTA): FTA é uma métrica em nível de modelo que avalia a correção de modelos identificados em logs analisados. Ela se baseia na precisão e na recuperação dos modelos extraídos de mensagens de log. A FTA é calculada como a média harmônica de Precisão e Recuperação, que são computadas especificamente para os modelos. Um modelo é considerado corretamente identificado se, e somente se, o log analisado compartilha o mesmo modelo que o modelo de verdade fundamental, e todos os tokens do modelo analisado correspondem exatamente aos do modelo de verdade fundamental. Precisão = , Recuperação = Número de modelos analisados . Número de modelos corretamente identificados.

Número de modelos de verdade fundamental . Portanto, FTA = $2\times \frac{\text{Precisão}\times\text{Recuperação}}{\text{Precisão}+\text{Recuperação}}$. onde o FTA fornece uma medida equilibrada de precisão de identificação de modelos, considerando tanto a capacidade de identificar modelos corretamente (precisão) quanto a capacidade de capturar todos os modelos de verdade básica (recall).

IV. RESULTADOS DO ESTUDO

Nesta seção, apresentamos os resultados experimentais e fornecer uma análise para abordar cada questão de pesquisa.

A. Eficácia de dez analisadores baseados em LLM (RQ1)

Nesta seção, avaliamos o desempenho de dez analisadores sintáticos leves de código aberto baseados em LLM, bem como do modelo de referência de parâmetros grandes GPT-3.5-Turbo, em seis conjuntos de dados distintos que representam uma gama diversificada de dados de log. Conforme mostrado na Tabela III, avaliamos seu desempenho usando as métricas PA e FTA, que medem a capacidade dos modelos de substituir variáveis dinâmicas pelos modelos corretos em vários formatos de log.

No geral, o desempenho dos modelos apresenta variabilidade significativa entre os diferentes conjuntos de dados. O LHSB produz os melhores resultados na maioria dos modelos, o que pode sugerir que o conjunto de dados foi bem exposto aos modelos durante o treinamento.
Essa observação pode apontar para um potencial problema de vazamento de dados, onde os modelos já encontraram logs semelhantes ou os logs são baseados em formatos mais antigos com os quais os modelos estão mais familiarizados. Em contraste, o conjunto de dados FALL, que representa o conjunto de amostras mais diverso e complexo, resultou no menor desempenho em todos os modelos. Isso destaca os desafios impostos por conjuntos de dados com maior variabilidade e estruturas de log diversas, onde os modelos lutam para generalizar além de seus dados de treinamento. Além disso, o modelo de referência de grandes parâmetros GPT-3.5-Turbo não exibe uma vantagem clara sobre os analisadores leves baseados em LLM. Suas pontuações PA e FTA são apenas 2,5% e 2,7% maiores, respectivamente, do que as do modelo leve de melhor desempenho, Deepseek-R1-1.5B.

As diferenças mais notáveis aparecem nos conjuntos de dados LHSB e ILMSD, ambos com pontuações gerais mais altas em comparação aos outros quatro conjuntos de dados, indicando que o GPT-3.5-Turbo tem um desempenho relativamente melhor em ambientes de registro mais estruturados ou menos diversos, mas sua vantagem diminui em cenários complexos ou de poucos recursos.

Entre os modelos testados, Deepseek-R1-1.5B e Qwen-2.5-7B apresentam consistentemente o melhor desempenho na maioria dos conjuntos de dados. Especificamente, Qwen-2.5-7B exibe um comportamento único no Conjunto de dados EAILD, onde a pontuação FTA excede a pontuação PA. Essa anomalia sugere que, embora o Qwen-2.5-7B seja excelente na identificação de modelos, ele ainda pode enfrentar dificuldades na substituição precisa de variáveis dinâmicas nos logs. Isso indica que o modelo pode estar identificando modelos com mais precisão, mas ainda não substituindo todos os componentes dinâmicos com a precisão esperada. O Deepseek-R1-1.5B, por outro lado, apresenta desempenho robusto e equilibrado, com altas pontuações tanto em PA quanto em FTA, o que sugere que ele lida com a tarefa de análise de logs de forma mais eficaz em diversos formatos de log.

As diferenças de desempenho entre conjuntos de dados destacam que os analisadores sintáticos baseados em LLM não apresentam fortes capacidades de generalização, especialmente em condições de zero-shot. Embora os modelos tenham um bom desempenho em conjuntos de dados de referência como o LHSB, eles lutam para manter um desempenho consistente ao analisar logs de conjuntos de dados mais diversos e complexos, como o FALL. Isso sugere que, em configurações de zero-shot, os modelos dependem fortemente de conhecimento prévio de logs semelhantes ou de formatos desatualizados que encontraram durante o treinamento. Como resultado, os modelos podem não conseguir generalizar bem para estruturas de log novas e mais complexas, indicando uma limitação em sua capacidade de se adaptar a novos dados de log sem aprendizado adicional de poucos disparos ou treinamento específico de domínio. Isso também ressalta a importância de melhorar a capacidade de generalização dos analisadores sintáticos baseados em LLM, particularmente para tarefas de análise de log do mundo real, onde os formatos de log podem variar muito.

Resposta à RQ1: A avaliação dos dez analisadores de LLM de código aberto em seis conjuntos de dados revela uma variabilidade significativa no desempenho, com modelos apresentando melhor desempenho no LogHub Subset Benchmark (LHSB) e dificuldades com os logs mais diversos e complexos no conjunto de dados FALL.

B. Estudo de Ablação de Diferentes Fatores (RQ2)

Nesta seção, examinamos o impacto das características do log no desempenho da análise, concentrando-nos em dois fatores: o comprimento das entradas do log e a frequência das variáveis dinâmicas.
Critérios detalhados de classificação podem ser encontrados na Seção III-D.
Categorizamos os logs dos seis conjuntos de dados em três grupos com base no comprimento da entrada do log: logs curtos, logs de comprimento médio e logs longos. Os resultados correspondentes são apresentados na Tabela IV. Além disso, classificamos os logs pela frequência de variáveis dinâmicas em três grupos: logs com 1–2 variáveis, 3–4 variáveis e 5 ou mais variáveis. Os resultados para esse agrupamento são apresentados na Tabela V.

No geral, o comprimento das entradas de log impacta significativamente o desempenho da análise. Logs curtos e médios apresentam desempenho relativamente semelhante entre os modelos, com os curtos superando ligeiramente os médios. De fato, para logs curtos, a maioria dos modelos apresenta desempenho superior aos resultados obtidos no RQ1, sugerindo que logs mais curtos podem ser mais fáceis de manipular para analisadores baseados em LLM. No entanto, o desempenho de todos os modelos apresenta um declínio consistente ao processar logs longos, com uma queda mais perceptível no desempenho do conjunto de dados FALL. Esse declínio provavelmente se deve à maior complexidade e variabilidade de logs mais longos, que apresentam mais desafios para identificar e substituir corretamente variáveis dinâmicas. Logs

TABELA III: Desempenho de dez analisadores LLM de código aberto em seis conjuntos de dados (%).

Modelo	LHSB		DSL		LSWAL		EAILD		ILMSD		CAIR		Média	
	PA	FTA	PA	FTA	PA	FTA	PA	FTA	PA	FTA	PA	FTA	PA	FTA
Deepseek-R1-1.5B	**80,2**	76,7	**76,3**	68,4	**74,9**	**70,3**	**72,5**	66,7	**77,3**	**74,8**	**68,4**	**62,5**	**74,5**	69,9
Gemma-2-2B	65,2	60,8	61,3	56,2	61,2	56,1	54,3	53,2	65,2	59,8	53,7	49,1	60,2	55,9
Gemma-2-9B	74,2	72,1	70,7	65,3	70,1	66,9	65,2	62,8	72,8	70,4	63,7	58,6	69,5	66,0
Llama-3.2-1B	66,6	63,2	65,4	59,8	63,7	56,9	58,0	56,8	70,2	62,4	56,9	50,4	63,5	58,3
Llama-3.2-3B	62,7	57,2	58,9	52,5	59,4	54,6	52,8	50,4	63,6	57,9	50,2	46,7	58,0	53,2
Llama-3.1-8B	70,6	67,3	67,5	62,1	68,4	61,7	60,1	60,2	71,2	67,9	60,5	53,8	66,4	62,2
Qwen-2.5-0.5B	61,3	56,8	56,3	51,2	57,9	52,4	50,2	50,3	62,2	58,3	48,3	41,6	56,0	51,8
Qwen-2.5-1.5B	68,2	64,1	64,9	59,3	65,2	59,1	57,2	57,9	68,5	63,6	58,1	51,2	63,7	59,2
Qwen-2.5-3B	73,9	72,8	71,2	66,8	69,4	67,5	63,1	64,5	70,2	68,6	61,1	57,0	68,2	66,2
Qwen-2.5-7B	78,5	**77,4**	74,6	**69,9**	72,5	70,2	68,8	**69,2**	74,6	73,1	66,0	60,2	72,5	**70,0**
Média	70,1	66,8	66,7	61,2	66,3	61,6	60,2	59,2	69,6	65,7	58,7	53.1	65,3	61,3
GPT-3.5 (linha de base)	83,7	80,5	77,4	71,2	76,1	70,9	74,4	70,4	80,6	78,2	69,9	64,0	77,0	72,6

no conjunto de dados FALL são mais diversos em termos de estrutura e conteúdo, agravando ainda mais as dificuldades de análise por mais tempo registros.

Da mesma forma, a frequência das variáveis dinâmicas desempenha um papel crítico papel no desempenho do modelo. Logs contendo cinco ou mais variáveis dinâmicas representam desafios significativos para todos os modelos, com desempenho geral inferior ao observado por longos registros. Isso é especialmente evidente no conjunto de dados FALL, onde o O FTA é de apenas 46,9%, o que representa uma redução de 6,2% em relação ao desempenho observado na Seção IV-A. Uma vez que um log é considerado corretamente analisados somente quando todos os componentes dinâmicos são identificados e substituídos com precisão, a presença de múltiplas variáveis aumenta dificuldade de análise significativamente. Este desafio é particularmente pronunciado ao lidar com logs com cinco ou mais dinâmicos variáveis, onde os modelos geralmente apresentam desempenho ruim em todas conjuntos de dados, incluindo modelos avançados como Deepseek-R1-1.5B. A dificuldade surge da complexidade de garantir substituições precisas para múltiplas variáveis dentro de uma única entrada de log, exigindo maior precisão dos modelos. Assim, pesquisas futuras devem se concentrar no desenvolvimento de métodos de análise especificamente adaptado para lidar efetivamente com entradas de log caracterizadas por alta complexidade e inúmeras variáveis dinâmicas.

Resposta à RQ2: O estudo de ablação mostra que o comprimento do log e a frequência das variáveis dinâmicas impactam significativamente a desempenho de analisadores baseados em LLM, com logs e logs mais longos contendo variáveis mais dinâmicas levando a um declínio perceptível na precisão da análise.

C. Impacto do aprendizado de poucos disparos no desempenho da análise de logs (RQ3)

Nesta seção, examinamos o impacto do aprendizado de poucas tentativas sobre o desempenho de analisadores baseados em LLM em tarefas de análise de log. Para avaliar isso, ajustamos os modelos selecionados usando dois abordagens: (1) Prompting de poucos exemplos baseado em exemplos e (2) Prompting de Poucas Oportunidades Baseado em Regras de Substituição. A abordagem baseada em exemplos fornece ao modelo um conjunto de logs originais e seus modelos analisados correspondentes como demonstrações, enquanto a abordagem baseada em regras fornece regras de substituição explícitas para orientar o processo de análise. Conforme mostrado na Tabela VI, comparamos o desempenho do PA dos dois aparelhos ajustados métodos com os de suas contrapartes de tiro zero em todo o mesmos seis conjuntos de dados.

No geral, a aprendizagem de poucos tiros leva a melhorias significativas na análise do desempenho em todos os modelos, independentemente do método específico utilizado. Os modelos ajustados de forma consistente superar seus equivalentes de tiro zero em todos os conjuntos de dados, com um aumento médio de PA de 7,46%, destacando que a aprendizagem de poucos disparos permite que os modelos se adaptem de forma mais eficaz às nuances das tarefas de análise de log. Essa adaptação é especialmente perceptível em formatos de log complexos, como aqueles encontrados em três conjuntos de dados de poucos recursos, onde os modelos funcionam melhor entender a estrutura e o contexto das mensagens de log. Enquanto ambos os métodos de aprendizagem de poucos passos - baseados em exemplos (PA médio (melhoria: 7,71%) e baseado em regras (melhoria média de PA: 7,21%) — resultam em ganhos de desempenho; o método baseado em exemplos geralmente produz resultados ligeiramente superiores. a diferença entre os dois métodos não é substancial, sugerindo que ambas as abordagens ajudam efetivamente os modelos ajustar-se à tarefa. Após uma inspeção mais aprofundada, descobrimos que o método baseado em exemplos supera significativamente o método baseado em regras ao lidar com logs com cinco ou mais dinâmicos variáveis, indicando sua vantagem no processamento de variáveis mais complexas logs. Além disso, no conjunto de dados FALL, que contém mais amostras de log diversas e complexas, ambos métodos ajustados mostram melhorias substanciais, com PA média aumentando em 13,84%. Isso enfatiza a necessidade de aprendizagem de poucos tiros em Analisadores baseados em LLM, permitindo que o modelo faça análises mais precisas e substituições de modelos sensíveis ao contexto.

Resposta à RQ3: O aprendizado de poucos disparos melhora significativamente o log análise de desempenho em todos os modelos, especialmente em conjuntos de dados com maior diversidade de amostras. A abordagem baseada em exemplos supera o método baseado em regras, embora a diferença seja pequena.

V. IMPLICAÇÕES

Os resultados deste estudo fornecem insights significativos sobre a eficácia e as limitações do LLM leve baseado em analisadores sintáticos para análise de logs, particularmente no contexto de dados de log do mundo real. Uma implicação fundamental é que o desempenho de Os modelos de análise de log são altamente dependentes das características das entradas de log, como o comprimento do log e a frequência de

TABELA IV: Desempenho de dez analisadores LLM de código aberto em termos de comprimento de entrada de log (%).

Modelo	Comprimento	LHSB		DSL		LSWAL		EAILD		ILMSD		CAIR		Média	
		PA	FTA	PA	FTA	PA	FTA	PA	FTA	PA	FTA	PA	FTA	PA	FTA
Deepseek-R1-1.5B	Curto	**83,6**	**80,2**	**79,9**	**75,1**	**77,3**	**74,8**	75,0	70,2	**80,6**	**77,5**	**78,8**	**76,3**	**76,2**	**76,6**
	Médio	**80,7**	**78,5**	77,1	**75,8**	68,6	**73,7**	70,8	**71,3**	66,0	**76,4**	**72,9**	**71,4**	**64,2**	**71,5**	**70,4**
	Longo			75,1	**73,6**	66,8	**71,3**	**69,1**	**70,0**	64,4	**73,6**	**72,9**	**62,8**	**59,1**	**67,4**	**66,7**
Gemma-2-2B	Curto	67,4	62,9	63,7	60,9	64,6	59,4	57,3	55,8	67,5	62,9	64,1	61,7	61,2	62,2
	Médio	66,9	61,4	62,8	58,5	60,7	57,2	55,7	54,1	62,8	58,0	56,8	52,1	56,7	56,7
	Longo	63,8	57,6	58,3	54,1	57,4	52,9	51,7	50,6	62,7	57,3	48,2	44,3	53,9	52,8
Gemma-2-9B	Curto	77,6	74,9	73,2	69,8	73,0	68,5	68,3	64,9	74,4	72,8	72,5	70,3	70,2	70,9
	Médio	75,1	71,8	73,2	71,3	65,1	68,6	65,3	66,7	63,4	74,2	69,8	65,1	60,0	67,4	66,7
	Longo			70,6	68,3	62,8	67,2	63,1	62,9	60,2	70,2	68,8	59,1	55,7	62,9	62,8
Lhama-3.2-1B	Curto	68,3	65,8	67,9	64,1	65,8	59,4	60,0	57,3	72,8	64,9	65,8	63,5	63,4	64,6
	Médio	67,2	64,1	66,7	60,4	64,3	57,6	56,4	55,2	69,2	61,9	58,7	51,3	58,6	59,2
	Longo	63,8	61,6	62,7	56,4	61,9	54,3	55,7	53,9	68,2	60,6	50,3	47,2	57,2	56,2
Lhama-3.2-3B	Curto	63,9	60,2	61,7	58,2	63,1	58,0	54,6	53,9	66,4	60,9	59,7	58,3	58,8	59,7
	Médio	63,6	60,1	58,4	60,1	54,5	61,7	55,3	52,4	49,6	64,2	58,5	53,4	49,8	55,6	55,2
	Longo			55,7	55,2	50,3	56,8	53,2	50,1	47,6	60,6	54,0	44,1	40,3	50,0	49,4
Lhama-3.1-8B	Curto	72,6	68,8	69,3	66,9	70,3	62,9	62,5	61,8	74,6	69,2	68,4	66,8	66,6	67,9
	Médio	70,0	67,4	66,9	68,2	62,7	67,3	62,0	58,9	58,1	70,3	66,8	62,8	56,7	62,2	62,8
	Longo			64,9	65,2	60,3	65,7	58,6	57,4	57,1	68,4	66,0	52,8	49,3	60,1	58,9
Qwen-2,5-0,5B	Curto	63,5	58,2	57,4	55,9	59,6	54,8	53,4	52,0	65,7	60,9	57,3	53,9	56,9	57,8
	Médio	61,5	58,0	56,2	57,4	52,9	59,2	53,1	49,6	49,3	60,1	59,4	51,2	44,8	53,8	53.1
	Longo			53,4	52,2	49,3	54,1	49,5	47,3	48,0	59,3	54,0	42,5	38,4	48,4	48,4
Qwen-2,5-1,5B	Curto	70,2	66,6	67,2	64,7	67,3	62,9	60,4	58,5	69,9	65,8	66,7	64,3	64,0	64,9
	Médio	67,8	Longo	63,6	64,2	60,5	66,8	60,3	56,5	58,3	69,2	61,7	60,6	54,8	60,2	60,8
		65,2		62,5	62,7	57,3	63,1	56,9	54,8	54,1	66,4	61,7	53,1	48,6	57,8	57,0
Qwen-2.5-3B	Curto	75,6	74,7	74,5	71,3	72,0	69,1	64,8	65,8	72,6	70,2	69,7	68,3	68,6	69,2
	Médio	74,3	Longo	73,0	71,1	66,2	68,8	66,9	64,2	64,8	72,1	67,5	64,2	59,6	66,6	65,8
		70,8	Curto	70,1	69,4	64,5	67,2	65,1	60,5	62,3	68,4	65,9	55,4	54,1	61,7	61,3
Qwen-2.5-7B		81,4	80,0	77,8	74,5	76,3	73,7	**75,8**	**71,7**	78,1	76,9	75,7	73,9	75,3	75,3
	Médio	79,3	Longo	**77,7**	74,2	**69,0**	72,9	**71,5**	67,9	**68,5**	73,2	72,5	70,3	62,8	69,5	69,3
		76,1	Curto	**75,3**	72,2	**67,1**	70,3	67,6	66,2	**65,8**	71,9	72,4	60,3	57,6	**67,4**	65,8
Média		72,4	69,3	69,2	66,1	68,9	64,4	63,2	61,2	72,3	68,2	67,9	65,7	-	-
	Médio	70,6	Longo	67,2	67,2	61,8	66,4	62,0	60,0	58,7	69,2	64,9	61,5	55,6	-	-
		67,6		64,7	64,0	58,9	63,5	59,0	57,7	56,4	67,0	64,4	52,9	49,5	-	-

variáveis dinâmicas. Nossos resultados demonstram que o desempenho de analisadores sintáticos leves baseados em LLM é comparável ao do modelo de grandes parâmetros ChatGPT. Embora esses modelos apresentam bom desempenho em conjuntos de dados mais simples e uniformes, como LHSB, eles lutam com registros mais complexos e diversos, como aqueles no conjunto de dados FALL. Isso destaca a necessidade de melhorar a capacidade de generalização dos analisadores baseados em LLM, como seus o desempenho atual é limitado pelos desafios impostos por sistemas de software modernos, que envolvem tecnologias mais diversas e formatos de log dinâmicos.

Outra implicação importante é o papel significativo do aprendizado de poucos disparos no aumento da precisão da análise sintática de logaritmos. Nosso estudo demonstra que a aprendizagem de poucas tentativas, particularmente o método baseado em exemplos, leva a melhorias substanciais no desempenho do modelo em comparação com as configurações de zero tentativas. Esta descoberta destaca a importância de adaptar modelos a logs específicos tipos, já que o aprendizado de poucos disparos permite que os modelos se adaptem melhor a a estrutura e o conteúdo dos logs em cenários do mundo real. Por ilustrando a eficácia da aprendizagem de poucos tiros, este estudo sublinha que os modelos genéricos podem não ter um desempenho ideal em tarefas especializadas como análise de log, onde a diversidade e a complexidade dos dados de log exige abordagens mais direcionadas.

Por fim, o nosso estudo também destaca os desafios colocados por logs com alto número de variáveis dinâmicas, que dificultam significativamente o desempenho do modelo. Isso reforça a importância de entender como a complexidade dos dados de log impacta a precisão da análise e aponta para a necessidade crítica para modelos lidarem com entradas de log diversas e altamente variáveis. Estas descobertas contribuem para o crescente corpo de literatura sobre análise de log, ressaltando a necessidade de desenvolver mais sistemas de análise de log robustos e precisos, capazes de abordar as complexidades dos dados de log modernos.

VI. AMEAÇAS À VALIDADE

A. Qualidade e representatividade dos dados

O conjunto de dados LHSB pode introduzir vieses devido à sua relativa formato de log desatualizado, que os modelos podem já ter encontrados durante o pré-treinamento. Isso pode levar a alto desempenho neste conjunto de dados, especialmente quando comparado para outros como o FALL, que contém mais diversidade e complexidade entradas de log. Para mitigar essa ameaça, utilizamos vários conjuntos de dados de domínios variados para melhor avaliar a capacidade dos modelos de generalizar entre diferentes formatos de log e ambientes.

B. Generalização do método de aprendizagem de poucos disparos

As duas abordagens de aprendizagem de poucos passos - baseadas em exemplos e regras de substituição baseadas em regras — podem não abranger todas as possibilidades estratégias para adaptar modelos a tarefas de análise de log. Embora ambos

TABELA V: Desempenho de dez analisadores LLM de código aberto em termos de frequência de variáveis dinâmicas (%).

Modelo	Variável	LHSB		DSL		LSWAL		EAILD		ILMSD		CAIR		Média	
		PA	FTA	PA	FTA	PA	FTA	PA	FTA	PA	FTA	PA	FTA		
Deepseek-R1-1.5B	1-2	**84,9**	82,5	**81,2**	**76,0**	**79,6**	**75,4**	77,1	73,5	**82,4**	**79,3**	**80,2**	**77,9**	**80,9**	**80,1**
	3-4	80,3	77,6	**75,2**	67,3	**73,8**	71,4	**70,0**	65,2	**75,8**	**72,0**	**70,8**	65,9	**74,3**	**71,8**
	ÿ 5	**76,3**	73,8	**71,5**	64,9	**69,7**	**66,9**	**67,2**	62,0	**71,5**	**70,2**	**59,4**	**56,8**	**69,3**	**65,4**
Gemma-2-2B	1-2	69,8	63,7	65,2	62,5	65,9	60,8	59,2	58,1	69,4	63,6	67,3	62,9	66,1	65,9
	3-4	65,2	60,6	61,4	57,2	59,4	56,6	54,7	53,2	63,5	60,4	55,2	51,4	59,9	58,1
	ÿ 5	60,2	54,5	56,4	52.	55,2	50,7	49,6	48,3	60,5	55,9	45,7	40,9	54,6	51,5
Gemma-2-9B	1-2	78,3	75,6	75,4	71,3	76,2	70,5	70,9	66,7	75,9	74,0	75,8	73,1	75,4	74,8
	3-4	74,7	73,1	70,9	66,2	69,4	66,2	65,4	62,9	73,9	69,2	64,8	59,2	69,9	67,4
	ÿ 5	68,3	64,7	64,3	60,2	65,5	61,9	60,7	58,1	68,7	65,3	56,4	51,9	64,0	61,3
Lhama-3.2-1B	1-2	70,3	66,3	69,7	65,9	66,4	60,7	64,2	60,1	74,9	66,3	68,9	66,5	69,1	69,1
	3-4	67,0	64,8	67,4	60,0	64,1	57,2	55,6	53,7	67,1	60,3	57,2	50,3	63,1	59,6
	ÿ 5	61,4	59,9	60,2	55,8	59,5	52,7	53,1	50,2	66,4	58,2	48,5	45,3	58,2	55,0
Lhama-3.2-3B	1-2	66,8	62,4	65,3	60,7	65,9	60,4	57,9	55,4	69,2	64,4	63,5	61,6	64,8	64,7
	3-4	63,0	57,9	60,4	54,2	62,0	56,1	51,2	50,4	63,1	57,2	55,1	50,5	59,1	57,0
	ÿ 5	57,8	52,5	53,6	50,6	54,2	51,7	49,3	46,0	58,4	51,7	42,9	38,6	52,7	48,9
Lhama-3.1-8B	1-2	74,7	70,1	72,4	68,6	72,5	64,7	64,9	64,2	77,4	71,3	70,6	68,9	72,1	72,1
	3-4	70,2	67,3	67,6	61,4	65,2	61,9	57,3	56,4	71,1	67,3	61,4	55,7	65,5	64,2
	ÿ 5	64,9	62,5	63,8	57,5	63,1	56,3	55,3	54,0	66,2	64,5	50,5	47,0	60,6	57,8
Qwen-2,5-0,5B	1-2	65,7	60,4	59,2	56,6	60,3	55,9	55,7	53,6	62,3	62,8	59,4	54,2	60,4	59,8
	3-4	60,3	55,7	56,3	51,2	58,4	52,6	48,2	48,5	59,7	57,2	51,4	45,2	55,7	53,8
	ÿ 5	56,5	52,6	51,5	47,8	53,1	48,2	46,4	47,3	58,1	53,2	41,0	37,1	51,1	48.1
Qwen-2,5-1,5B	1-2	71,9	67,3	69,4	66,2	68,7	63,6	62,8	60,1	70,8	66,9	68,8	66,5	68,7	68,4
	3-4	66,8	62,7	63,9	59,4	64,5	59,8	57,4	58,8	70,3	62,5	60,4	53,6	63,9	62,1
	ÿ 5	63,1	60,5	60,3	54,8	62,0	54,1	52,1	51,8	63,2	59,3	50,1	46,5	58,5	55,5
Qwen-2.5-3B	1-2	77,9	75,8	76,3	74,1	74,2	70,6	67,2	68,0	74,5	72,9	71,3	70,0	73,6	72,5
	3-4	73,3	72,1	70,3	67,9	69,2	67,4	66,0	66,0	73,5	69,8	65,1	59,4	69,6	67,5
	ÿ 5	68,8	67,9	66,8	63,2	64,1	63,7	58,2	60,4	65,9	62,1	52,8	51,4	62,8	59,0
Qwen-2.5-7B	1-2	83,1	**82,7**	79,3	75,8	77,9	75,0	**78,0**	**77,5**	80,1	78,2	77,3	75,2	79,3	78,1
	3-4	**80,7**	**78,4**	**75,2**	**69,8**	73,6	**72,2**	66,3	**67,8**	72,7	71,9	**70,8**	**67,2**	73,2	71,2
	ÿ 5	74,3	**74,0**	70,5	**66,4**	67,2	64,8	63,5	**64,1**	68,1	68,0	57,8	55,2	66,9	63,2
Média	1-2	73,1	71,2	69,9	66,0	68,8	65,5	63,2	63,7	70,8	67,8	62,9	59,5	-	-
	3-4	70,2	67,0	66,9	61,5	66,0	62,1	59,2	58,3	69,1	64,8	61,2	55,8	-	-
	ÿ 5	65,2	62,3	61,9	57,3	61,4	57,1	55,5	54,2	64,7	60,8	50,5	46,9	-	-

TABELA VI: Desempenho de PA dos dois métodos de ajuste fino em comparação com suas contrapartes de disparo zero (%).

Modelo	LHSB		DSL		LSWAL		EAILD		ILMSD		CAIR	
	EB	RR-B	EB	RR-B	EB	RR-B	EB	RR-B	EB	RR-B	EB	RR-B
Deepseek-R1-1.5B	**85,6 (ÿ 5,4)** 84,8 (ÿ 4,6)		81,8 (ÿ 4,8)	80,7 (ÿ 4,4)	79,5 (ÿ 4,6)	72,4 (ÿ 7,2)	79,2 (ÿ 4,3)	81,3 (ÿ 8,8)	80,6 (ÿ 8,1)	67,1 (ÿ 5,9)	84,2 (ÿ 6,9)	83,7 (ÿ 6,4)	73,7 (ÿ	81,0 (ÿ 12,6) 79,9 (ÿ 11,5)
Gemma-2-2B	71,5 (ÿ 6,3) 67,4 (ÿ 6,1)		66,9 (ÿ 5,6)	67,6 (ÿ 6,4)	78,8 (ÿ 4,6)	77,9 (ÿ 3,7)	74,3 (ÿ 3,9)	62,5 (ÿ 8,2)	62,3 (ÿ 8,0)	74,1 (ÿ 4,0)	71,9 (ÿ 6,7)	8,5) 73,4 (ÿ 8,2)	78,5 (ÿ 5,7)	72,2 (ÿ 18,5) 71,5 (ÿ 17,8)
Gemma-2-9B	74,2 (ÿ 3,8) 74,3 (ÿ 4,2)		74 (ÿ 7,4)	73,1 (ÿ 6,5)	70,7 (ÿ 5,3)	70,4 (ÿ 5,0)	69,6 (ÿ 5,9)	71,3 (ÿ 6,1)	69,1 (ÿ 5,4)	65,5 (ÿ 7,5)	65,1 (ÿ 7,1)	77,9 (ÿ 5,1)	76,6 (ÿ 6,4)	76,4 (ÿ 76,4 (ÿ 12,7) 75,9 (ÿ 12,2)
Lhama-3.2-1B	(ÿ 6,2) 68,2 (ÿ 5,5)		64,2 (ÿ 5,3)	64,0 (ÿ 5,1)	64,5 (ÿ 5,1)	75,2 (ÿ 4,6)	74,7 (ÿ 4,1)	64,1 (ÿ 4,7)	60,2 (ÿ 7,4)	59,6 (ÿ 6,8)	72,4 (ÿ 4,0)	6,2) 70,5 (ÿ 6,9)	70,4 (ÿ 6,8)	70,2 (ÿ 13,3) 69,6 (ÿ 12,7)
Lhama-3,2-3B	4,0) 71,1 (ÿ 3,6)		72,9 (ÿ 4,5)	68,1 (ÿ 6,8)	67,2 (ÿ 5,9)	61,9 (ÿ 5,6)	61,2 (ÿ 4,9)	67,1 (ÿ 7,0)	67,3 (ÿ 7,2)	63,3 (ÿ 5,4)	58,4 (ÿ 8,2)	76,4 (ÿ 5,2)	76,5 (ÿ 5,3)	68,6 (ÿ 64,1 (ÿ 13,9) 63,4 (ÿ 13,2)
Lhama-3,1-8B	6,5) 75,3 (ÿ 7,1)		74,9 (ÿ 6,7)	72,1 (ÿ 7,2)	71,5 (ÿ 6,6)	71,4 (ÿ 6,2)	80,2 (ÿ 6,3)	58,0 (ÿ 7,8)	70,9 (ÿ 5,7)	66,1 (ÿ 8,9)	65,6 (ÿ 8,4)	6,4) 68,4 (ÿ 6,2)	75,5 (ÿ 7,0)	75,0 (ÿ 14,5) 74,9 (ÿ 14,4)
Qwen-2,5-0,5B	5,7) 77,3 (ÿ 6,1)		76,9 (ÿ 5,7)	75,2 (ÿ 5,8)	86,0 (ÿ 7,5)	85,8 (ÿ 7,3)	80,6 (ÿ 6,0)	74,6 (ÿ 5,2)	72,2 (ÿ 9,1)	71,5 (ÿ 8,4)	77,1 (ÿ 4,6)	75,0 (ÿ 6,5)	77,7 (ÿ 7,5)	77,2 (ÿ 60,8 (ÿ 12,5) 59,9 (ÿ 11,6)
Qwen-2,5-1,5B	5,6) 77,8 (ÿ 5,3) ÿ 5,5 ÿ 5,0 ÿ 5,5							78,4 (ÿ 9,6)	77,8 (ÿ 9,0) ÿ 8,1			7,0) 82,3 (ÿ 7,7)	81,8 (ÿ 7,2) ÿ	71,8 (ÿ 13,7) 71,2 (ÿ 13,1)
Qwen-2,5-3B												6,8 ÿ 6,5		75,6 (ÿ 14,5) 75,1 (ÿ 14,0)
Qwen-2,5-7B														81,3 (ÿ 15,3) 80,7 (ÿ 14,7)
Média. Melhoria	ÿ 6.3		ÿ 5.4		ÿ 4.9		ÿ 7.7					ÿ 14,2 ÿ 13,5		

os métodos demonstram ganhos de desempenho, sua eficácia podem variar entre domínios ou com conjuntos de dados além daqueles examinados neste estudo. Além disso, as melhorias observadas de A aprendizagem de poucos disparos pode ser altamente dependente do conjunto de dados. Além disso experimentação com conjuntos de dados adicionais e mais diversos é necessário avaliar mais detalhadamente a robustez e escalabilidade dessas abordagens.

VII. CONCLUSÃO

Este estudo empírico avalia o desempenho de dez analisadores LLM leves de código aberto com parâmetros pequenos tamanhos em seis conjuntos de dados cuidadosamente construídos, representando ambientes de registro modernos de grande escala e de poucos recursos. Nossos resultados revelam uma variabilidade significativa em PA e FTA, com modelos que alcançam médias de até 70,1% PA e 66,8% FTA em conjuntos de dados de referência como LHSB, mas caindo para 58,7% PA e 53,1% FTA em conjuntos de dados mais diversos, como FALL. Registros mais longos e aqueles que contêm variáveis mais dinâmicas também resultou em declínios perceptíveis no desempenho. A aprendizagem em poucas tentativas — particularmente a abordagem baseada em exemplos — demonstrou melhorias substanciais, alcançando o maior ganho de 14,2% PA no conjunto de dados mais desafiador, FALL. Esses resultados ressaltam a importância da diversidade dos conjuntos de dados na avaliação comparativa e destacar o potencial da aprendizagem direcionada de poucos tiros para melhorar substancialmente as capacidades de generalização dos LLMs em tarefas de análise de log.

REFERÊNCIAS

[1] X. Wu, H. Li e F. Khomh, “Sobre a eficácia da representação de log para detecção de anomalias baseada em log”, Empirical Software Engineering, vol. 28, no. 6, p. 137, 2023.

[2] Z. Li, C. Luo, T.-H. Chen, W. Shang, S. He, Q. Lin e D. Zhang, “Perdemos algo importante? estudando e explorando a abstração de log com reconhecimento de variáveis”, em 2023 IEEE/ACM 45ª Conferência Internacional de Engenharia de Software (ICSE). IEEE, 2023, pp. 830–842.

[3] X. Xie, S. Jian, C. Huang, F. Yu e Y. Deng, “Logrep: Detecção de anomalias baseada em log representando informações semânticas e numéricas em mensagens brutas”, em 2023 IEEE 34º Simpósio Internacional sobre Engenharia de Confiabilidade de Software (ISSRE). IEEE, 2023, pp. 194–206.

[4] Y. Sun, J. Keung, J. Zhang, HK Yu, W. Luo e S. Liu, “Revelando anomalias ocultas: Aproveitando o smac-lstm para análise aprimorada de logs de software”, na 48ª Conferência Anual de Computadores, Software e Aplicações (COMPSAC) do IEEE de 2024. IEEE, 2024, pp. 1178–1183.

[5] B. Yu, J. Yao, Q. Fu, Z. Zhong, H. Xie, Y. Wu, Y. Ma e P. He, “Aprendizado profundo ou aprendizado de máquina clássico? Um estudo empírico sobre detecção de anomalias baseada em logaritmos”, em 2024, IEEE/ACM 46ª Conferência Internacional de Engenharia de Software (ICSE). Sociedade de Computação IEEE, 2023, pp. 392–404.

[6] V.-H. Le e H. Zhang, “Detecção de anomalias baseada em log com aprendizado profundo: até onde estamos?” em Anais da 44ª conferência internacional sobre engenharia de software, 2022, pp. 1356–1367.

[7] ZA Khan, D. Shin, D. Bianculli e L. Briand, “Diretrizes para avaliar a precisão das técnicas de identificação de modelos de mensagens de log”, em Anais da 44ª Conferência Internacional de Engenharia de Software, 2022, pp. 1095–1106.

[8] J. Zhu, S. He, J. Liu, P. He, Q. Xie, Z. Zheng e MR Lyu, “Ferramentas e benchmarks para análise automatizada de logs”, em 2019 IEEE/ACM 41ª Conferência Internacional sobre Engenharia de Software: Engenharia de Software na Prática (ICSE-SEIP). IEEE, 2019, pp. 121–130.

[9] P. He, J. Zhu, Z. Zheng e MR Lyu, “Drain: Uma abordagem de análise de log online com árvore de profundidade fixa”, na conferência internacional IEEE de 2017 sobre serviços web (ICWS). IEEE, 2017, pp. 33–40.

[10] H. Dai, H. Li, C.-S. Chen, W. Shang e T.-H. Chen, “Logram: análise de log eficiente usando dicionários n-gram”, IEEE Transactions on Software Engineering, vol. 48, nº 3, pp. 879–892, 2020.

[11] M. Du e F. Li, “Spell: Análise de streaming de logs de eventos do sistema”, em 2016 IEEE 16ª Conferência Internacional sobre Mineração de Dados (ICDM). IEEE, 2016, págs. 859–864.

[12] X. Wang, X. Zhang, L. Li, S. He, H. Zhang, Y. Liu, L. Zheng, Y. Kang, Q. Lin, Y. Dang et al., “Spine: um analisador de log escalável com orientação de feedback”, em Anais da 30ª Conferência Europeia Conjunta de Engenharia de Software da ACM e Simpósio sobre os Fundamentos da Engenharia de Software, 2022, pp. 1198–1208.

[13] Z. Jiang, J. Liu, Z. Chen, Y. Li, J. Huang, Y. Huo, P. He, J. Gu e MR Lyu, “Lilac: Análise de log usando llms com cache de análise adaptável”, Anais da ACM em Engenharia de Software, vol. 1, nº FSE, pp. 137–160, 2024.

[14] V.-H. Le e H. Zhang, “Análise de log com aprendizagem de poucos disparos baseada em prompts”, em 2023 IEEE/ACM 45ª Conferência Internacional sobre Engenharia de Software (ICSE). IEEE, 2023, pp. 2438–2449.

[15] Z. Xu, B. Huang e N. Chen, “Análise de log usando aprendizado de prompt baseado em filtragem semântica”, em 2024 IEEE 24ª Conferência Internacional sobre Qualidade, Confiabilidade e Segurança de Software (QRS). IEEE, 2024, pp. 667–676.

[16] Z. Ma, AR Chen, DJ Kim, T.-HP Chen e S. Wang, “Llmparser: Um estudo exploratório sobre o uso de grandes modelos de linguagem para análise de logs”, 2024.

[17] J. Zhu, S. He, P. He, J. Liu e MR Lyu, “Loghub: Uma grande coleção de conjuntos de dados de log do sistema para análise de log orientada por IA”, em 2023 IEEE 34º Simpósio Internacional sobre Engenharia de Confiabilidade de Software (ISSRE). IEEE, 2023, págs. 355–366.

[18] M. Landauer, F. Skopik e M. Wurzenberger, “Uma revisão crítica de conjuntos de dados de log comuns usados para avaliação de técnicas de detecção de anomalias baseadas em sequência”, Proceedings of the ACM on Software Engineering, vol. 1, nº FSE, pp. 1354–1375, 2024.

[19] Y. Sun, JW Keung, Z. Yang, S. Liu e HK Yu, “Semirald: Um modelo de linguagem híbrida semi-supervisionado para detecção robusta de logs anômalos”, Tecnologia da Informação e Software, p. 107743, 2025.

[20] B. Yu, J. Yao, Q. Fu, Z. Zhong, H. Xie, Y. Wu, Y. Ma e P. He, “Aprendizagem profunda ou aprendizagem de máquina clássica? Um estudo empírico sobre detecção de anomalias baseada em log”, em Anais da 46ª Conferência Internacional IEEE/ACM sobre Engenharia de Software, 2024, pp. 1–13.

[21] C. Almodovar, F. Sabrina, S. Karimi e S. Azad, “Logfit: Detecção de anomalias de log usando modelos de linguagem ajustados”, IEEE Transactions on Network and Service Management, vol. 21, nº 2, pp. 1715–1723, 2024.

[22] A. Tufek e MS Aktas, “Sobre as técnicas de extração de proveniência de arquivos de log em grande escala”, Concurrency and Computation: Practice and Experience, vol. 35, no. 15, p. e6559, 2023.

[23] Y. Lin e Y.-Y. Chiang, “Uma abordagem de aprendizagem semi-supervisionada para previsão de eventos anormais em dados de séries temporais de operação de grandes redes”, na Conferência Internacional IEEE sobre Big Data de 2022 (Big Data). IEEE, 2022, pp. 1024–1033.

[24] J. Xu, R. Yang, Y. Huo, C. Zhang e P. He, “Divlog: análise de log com aprendizado em contexto aprimorado por prompt”, em Anais da 46ª Conferência Internacional IEEE/ACM sobre Engenharia de Software, 2024, pp. 1–12.

[25] KI Roumeliotis e ND Tselikas, “Chatgpt e modelos de IA aberta: uma revisão preliminar”, Future Internet, vol. 15, no. 6, p. 192, 2023.

[26] J. White, S. Hays, Q. Fu, J. Spencer-Smith e DC Schmidt, “Padrões de prompt do Chatgpt para melhorar a qualidade do código, refatoração, elicitação de requisitos e design de software”, em IA Generativa para Desenvolvimento Eficaz de Software. Springer, 2024, pp. 71–108.

[27] S. Gao, X.-C. Wen, C. Gao, W. Wang, H. Zhang e MR Lyu, “O que torna boas demonstrações em contexto para tarefas de inteligência de código com llms?” em 2023, 38ª Conferência Internacional IEEE/ACM sobre Engenharia de Software Automatizada (ASE). IEEE, 2023, pp. 761–773.

[28] J. Qi, S. Huang, Z. Luan, S. Yang, C. Fung, H. Yang, D. Qian, J. Shang, Z. Xiao e Z. Wu, “Loggpt: Explorando o chatgpt para detecção de anomalias baseada em log”, em 2023, Conferência Internacional IEEE sobre Computação e Comunicações de Alto Desempenho, Ciência de Dados e Sistemas, Cidade Inteligente e Confiabilidade em Sensores, Nuvem e Sistemas e Aplicações de Big Data (HPCC/DSS/SmartCity/DependSys). IEEE, 2023, pp. 273–280.

[29] W. Xu, L. Huang, A. Fox, D. Patterson e MI Jordan, “Detecção de problemas de sistema em larga escala por meio da mineração de logs de console”, em Anais do 22º simpósio ACM SIGOPS sobre princípios de sistemas operacionais, 2009, pp. 117–132.

[30] Q. Lin, H. Zhang, J.-G. Lou, Y. Zhang e X. Chen, “Identificação de problemas baseada em agrupamento de logs para sistemas de serviços online”, em Anais da 38ª Conferência Internacional sobre Engenharia de Software Compan-ion, 2016, pp. 102–111.

[31] M. Du, F. Li, G. Zheng e V. Srikumar, “Deeplog: detecção e diagnóstico de anomalias a partir de registros do sistema por meio de aprendizado profundo”, em Anais da conferência ACM SIGSAC de 2017 sobre segurança de computadores e comunicações, 2017, pp. 1285–1298.

[32] A. Oliner e J. Stearley, “O que dizem os supercomputadores: Um estudo de cinco registros de sistemas”, na 37ª conferência internacional anual IEEE/IFIP sobre sistemas e redes confiáveis (DSN'07). IEEE, 2007, pp. 575–584.

[33] N. Tremblay, S. Barthelmé e P.-O. Amblard, “Processos de pontos determinantes para conjuntos de cores”, Journal of Machine Learning Research, vol. 20, n.º 168, pp. 1–70, 2019.

[34] N. Reimers e I. Gurevych, “Sentence-bert: incorporações de frases usando redes bert siamesas”, pré-impressão arXiv arXiv:1908.10084, 2019.

[35] H. Abdi e LJ Williams, “Análise de componentes principais”, Wiley interdisciplinary reviews: computational statistics, vol. 2, no. 4, pp. 433– 459, 2010.

[36] P. Xia, L. Zhang e F. Li, “Aprendizagem de similaridade com conjunto de similaridade de cosseno”, Ciências da informação, vol. 307, pp. 39–52, 2015.

[37] X. Lin, W. Wang, Y. Li, S. Yang, F. Feng, Y. Wei e T.-S. Chua, “Ajuste fino com eficiência de dados para recomendação baseada em llm”, em Anais da 47ª conferência internacional ACM SIGIR sobre pesquisa e desenvolvimento em recuperação de informação, 2024, pp. 365–374.

[38] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi et al., “Deepseek-r1: Incentivar a capacidade de raciocínio em llms por meio de aprendizagem por reforço”, pré-impressão arXiv arXiv:2501.12948, 2025.

[39] G. Team, M. Riviere, S. Pathak, PG Sessa, C. Hardin, S. Bhupatiraju, L. Hussenot, T. Mesnard, B. Shahriari, A. Ramé et al., “Gemma 2: Melhorando modelos de linguagem aberta em um tamanho prático”, pré-impressão arXiv arXiv:2408.00118, 2024.

[40] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan et al., “The lhama 3 rebanho de modelos”, pré-impressão arXiv arXiv:2407.21783, 2024.

[41] A. Yang, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Li, D. Liu, F. Huang, H. Wei et al., “Relatório técnico Qwen2.5”, pré-impressão arXiv arXiv:2412.15115, 2024.

[42] Y. Zhang, K. Zhou e Z. Liu, “O que constitui bons exemplos para aprendizagem visual em contexto?” Advances in Neural Information Processing Systems, vol. 36, pp. 17 773–17 794, 2023.

[43] SK Singh, S. Kumar e PS Mehra, “Chat gpt & google bard ai: uma revisão”, em Conferência Internacional de 2023 sobre IoT, Comunicação e Tecnologia de Automação (ICICAT). IEEE, 2023, pp. 1–6.

[44] A. Kulesza e B. Taskar, “k-dpps: Processos de ponto determinante de tamanho fixo”, em Anais da 28ª Conferência Internacional sobre Aprendizado de Máquina (ICML-11), 2011, pp. 1193–1200.

[45] LE Peterson, “K-vizinho mais próximo”, Scholarpedia, vol. 4, n.º 2, p. 1883, 2009.