FedDALE: Detecção de Anomalias Eficiente em *Logs* Distribuídos com Pequenos Modelos de Linguagem Federados

Gabriel U. Talasso, Allan M. de Souza, Leandro A. Villas

[1]Universidade Estadual de Campinas (UNICAMP), Brasil,
g235078@dac.unicamp.br,
{allanms, lvillas}@unicamp.br

Resumo. *Com a crescente ubiquidade de sistemas distribuídos, falhas e ataques tornam-se cada vez mais frequentes, motivando o desenvolvimento de técnicas para mitigar essas vulnerabilidades. A análise de logs é uma abordagem promissora para detecção de anomalias, mas apresenta desafios importantes, relacionados a escala, natureza sequencial e textual dos dados, distribuição entre múltiplos dispositivos e informações sensíveis. Adicionalmente, dispositivos de baixo recurso existentes na redes são prejudicados pelos altos custos de treinamento, transmissão e uso de soluções atuais. Neste trabalho, apresentamos o FedDALE, um método federado e eficiente em comunicação e computação para detecção não-supervisionada de anomalias em logs utilizando pequenos modelos de linguagem (SLMs). A abordagem combina o (i) treinamento local não-supervisionado de modelos com ajuste fino eficiente, (ii) transferência de conhecimento federada baseada em predição e filtragem de dados, e (iii) treinamento de um modelo estudante menor e mais eficiente que agrega o conhecimento dos professores distribuídos. Experimentos, com implementação disponível*[1]*, mostram que o FedDALE atinge desempenho de detecção comparável a grandes modelos federados (F1 superior a 90%), enquanto reduz os custos de comunicação em até 82% e a latência de inferência em 71%.*

Abstract. *With the increasing ubiquity of distributed systems, failures and attacks have become more frequent, motivating the development of techniques to mitigate these vulnerabilities. Log analysis is a promising approach for anomaly detection, but it presents important challenges related to scale, the sequential and textual nature of the data, distribution across multiple devices, and sensitive information. Additionally, resource-constrained devices present in networks are hindered by the high costs of training, transmission, and deployment of current solutions. In this work, we present FedDALE, a federated and communication- and computation-efficient method for unsupervised anomaly detection in logs using Small Language Models (SLMs). The approach combines (i) unsupervised local training of models with parameter-efficient fine-tuning, (ii) federated knowledge transfer based on prediction and data filtering, and (iii) training of a smaller and more efficient student model that aggregates the knowledge of the distributed teachers. Experiments, with available code base, show that FedDALE achieves detection performance comparable to large federated models (F1 above 90%), while reducing communication costs by up to 82% and inference latency by 71%.*

1. Introdução

Atualmente, sistemas distribuídos fazem parte do cotiano através de uma ampla gama de aplicações envolvendo redes de computadores como *smartphones* e Internet das Coisas (IoT)

[1]Código disponível em: https://github.com/GabrielTalasso/FedDALE

até grandes servidores e *datacenters*, cada um deles presente de forma cada vez mais ubíqua na sociedade. No entanto, o amplo uso dessas tecnologias levanta alertas quanto à sua segurança, seja quanto a problemas e falhas no funcionamento dos sistemas como também em possíveis ataques maliciosos visando gerar prejuízos aos participantes e usuários das aplicações [Pang et al. 2021]. Dessa forma, em ambos os casos, o estudo para a detecção e monitoramento desses ataques e falhas, comumente chamados de forma geral de **anomalias**, é cada vez mais necessário para proteger de forma eficiente e performática esses sistemas computacionais [Li et al. 2022, Pang et al. 2021, Talasso et al. 2025].

Os *logs* são registros das operações e funções executadas durante o funcionamento dos sistemas e podem ser utilizados como uma opção para detecção de anomalias. Por representarem, de forma sequencial, contextual e ainda serem coletados em grande quantidade, esses dados de *logs* fornecem uma alternativa de valor para desenvolver métodos eficientes e robustos para detecção de anomalias. Entretanto, apresentam desafios devido a sua natureza complexa, textual e temporal [Xu et al. 2009]. Além disso, por estarem disponíveis em grandes quantidades, a criação de rótulos que identificam cada anomalia é extremamente custosa e dificultada.

Dessa forma, o uso de modelos de linguagem tem tomado evidência na área de detecção de anomalias em *logs*. Esses modelos são uma boa alternativa às técnicas clássicas de Aprendizado de Máquina para detecção uma vez que conseguem lidar melhor com os dados contextuais, textuais, sequenciais e complexos associados à coleta dos registros dos sistemas computacionais além de poderem ser treinados de forma auto-supervisionada, ou ainda não-supervisionada demandando menos rotulagem manual dos dados de treinamento [Guo et al. 2021, Almodovar et al. 2024, Guan et al. 2024, Talasso et al. 2025]. Porém, apesar de expressarem uma promissora alternativa quanto a qualidade das detecções, o uso e treinamento desses modelos é muito custoso para dispositivos com recursos limitados devido principalmente a quantidade de parâmetros necessários, principalmente quando se trata de dispositivos mais restritivos, como o caso de *smartphones* e IoT [Li et al. 2022].

Portanto, soluções mais eficientes têm sido propostas para mitigar problemas de performance associados ao treinamento de modelos de linguagem para detecção de anomalias. A primeira delas seria a utilização de Pequenos Modelos de Linguagem (*Small Language Models* - SLMs) [Wang et al. 2024] que funcionam como alternativa de menor tamanho, em número de parâmetros, aos Grandes Modelos de Linguagem (*Large Language Models* - LLMs), assim diminuindo os custos associados ao treinamento e no momento da detecção (inferência) [Allal et al. 2024, Talasso et al. 2025]. Outra alternativa seria o uso de técnicas de ajuste eficiente, conhecidas como *Parameter Efficent Finetuning Techniques* - PEFT que treina apenas um subconjunto dos parâmetros para evitar sobrecargas de recursos no ajuste e utilização dos modelos [Hu et al. 2021, Ye et al. 2024, Almodovar et al. 2024].

Outro desafio inerente aos dados de *logs* de sistemas computacionais está relacionado a natureza distribuída desses dados, uma vez que são produzidos por diversas fontes de dados separadas. Além disso, os *logs*, por representarem o comportamento detalhado desses sistemas, são sensíveis, carregando informações muitas vezes confidenciais e passíveis de ataques [Xu et al. 2009, Li et al. 2022]. Dessa forma, técnicas promissoras como o Aprendizado Federado (*Federated Learning* - FL), que permite o treinamento distribuído, colaborativo e sem o compartilhamento de dados sensíveis, se tornam essenciais para avanços na qualidade da detecção de anomalias [Huong et al. 2022, Zhang et al. 2022, McMahan et al. 2017].

Desafios. Dessa forma, a criação de técnicas que lidem simultaneamente com *(i)* dados

de *logs* sequenciais, textuais e não rotulados, de natureza complexa, considerando também *(ii)* eficiência em recursos, demandando menos memória e processamentos dos dispositivos de borda, muitas vezes limitados, para o treinamento e para detecção das anomalias de forma rápida e eficiente e ainda assim considerem *(iii)* a natureza privada e distribuída dos *logs* dos sistemas computacionais, sem a demanda de grandes quantidades de dados privados, ainda representam uma lacuna na literatura e são alvo desse trabalho.

Contribuições. Com esses desafios em vista, propomos o FedDALE (**D**etecção de **A**nomalias em ***L****ogs* **E**ficiente), uma solução que promove o FL em conjunto com a utilização de modelos de linguagem para um treinamento mais eficiente e performático de detecção de anomalias. O FedDALE utiliza três componentes principais sendo *(i)* um treinamento completamente não-supervisionado, sem a necessidade de rótulos em nenhuma etapa do processo em união com *(ii)* SLMs e PEFT em conjunto com técnicas de transferência do conhecimento, para a criação de modelos pequenos e eficientes enquanto mantém uma alta performance na detecção e ainda *(iii)* utilizando o FL para permitir o acesso a mais dados distribuídos de qualidade enquanto aumenta a privacidade dos participantes. Dessa forma, as nossas contribuições com esses trabalhos podem ser resumidas como:

- Propomos o FedDALE, uma técnica de detecção de anomalias federada e eficiente completamente não-supervisionada, que promove o uso de SLMs para alta performance de detecção em *logs* textuais, sequenciais e sem rótulos.
- Diminuímos os custos e aumentamos a velocidade da detecção de anomalias propondo o uso de transferência do conhecimento para criação de modelos pequenos e efetivos em adição ao uso do FL para aumento da privacidade e acesso a mais dados.
- Diminuímos os custos de comunicação, quando comparados ao FL tradicional, por comunicar apenas predições em vez de modelos completos, permitindo o treinamento mais rápido e eficiente para dispositivos mais restritos.
- Apresentamos uma análise extensiva do FedDALE em ambientes e dispositivos reais demonstrando eficiência com redução de até 82% de comunicação e 71% no tempo de resposta de detecção enquanto mantém performances comparáveis a modelos maiores acima de 90% de F1-score.

2. Trabalhos Relacionados

Diversos trabalhos recentes exploram modelos de linguagem para detecção de anomalias em *logs*. O LogBERT [Guo et al. 2021] utiliza a arquitetura BERT [Devlin 2018] treinada do zero para modelar sequências de mensagens de *logs*, onde cada mensagem é previamente transformada em um *template*. Embora essa abordagem permita capturar padrões sequenciais nos dados, ela apresenta limitações importantes, como o alto custo computacional associado ao treinamento completo do modelo. De forma semelhante, LogFiT [Almodovar et al. 2024] emprega a arquitetura RoBERTa [Liu 2019] pré-treinada e realiza seu ajuste fino para aprender o comportamento normal dos *logs*. A detecção é realizada com base no erro de predição do modelo, assumindo que grandes desvios indicam possíveis anomalias, de maneira não-supervisionada. Embora o uso de modelos pré-treinados aumente a flexibilidade e reduza o custo em relação ao treinamento do zero, essa abordagem ainda exige o ajuste completo de todos os parâmetros do modelo, tornando seu custo ainda alto.

Outras abordagens exploram modelos generativos maiores ou cenários distribuídos. O LogLLM [Guan et al. 2024] formula o problema como uma tarefa supervisionada de classificação, combinando representações semânticas obtidas com BERT e um modelo Llama instruído em linguagem natural para decidir se uma sequência de *logs* é anômala. Apesar

de aumentar a flexibilidade da solução, o método depende de dados rotulados e do ajuste de modelos de grande porte, o que implica alto custo computacional. Alternativamente, o LogGPT [Qi et al. 2023] utiliza um modelo GPT acessado via API e emprega *prompting* com exemplos rotulados para realizar a classificação sem necessidade de treinamento adicional. Embora essa estratégia reduza o custo computacional local, ela apresenta desempenho inferior em termos de precisão e sensibilidade, além de levantar preocupações de privacidade devido ao envio de *logs* para serviços externos. Por fim, FedLog [Li et al. 2022] aborda o problema no contexto federado, propondo um modelo baseado em redes convolucionais temporais treinado colaborativamente entre dispositivos utilizando FL e técnicas de poda para reduzir o custo de comunicação. No entanto, a abordagem ainda requer o treinamento completo do modelo em todos os clientes, o que limita sua aplicabilidade em ambientes com recursos computacionais restritos.

De maneira similar, [Talasso et al. 2025] une o ajuste eficiente de grandes modelos com o FL para tornar possível uma detecção de anomalias distribuída menos custosa. Essa metodologia usa técnicas de ajuste eficiente em parâmetros, como os *Low Rank Adapters* (LoRA) [Hu et al. 2021] com FL [Ye et al. 2024] para permitir eficiência em recursos e acesso à dados distribuídos e privados. Ainda assim, a performance dos modelos propostos depende diretamente do custo associado ao treinamento e à comunicação desses modelos, ou seja, performances altas dependem de modelos grandes, limitando o uso para aplicações críticas que requerem baixa latência e alta performance simultaneamente.

Para mitigar os altos custos de comunicação e treinamento ou uso dos modelos em FL, diversos métodos recentes exploram estratégias mais eficientes, como *one-shot federated learning* e *knowledge distillation* federada. Em abordagens *one-shot* [Guha et al. 2019, Li et al. 2020a], os clientes treinam modelos locais de forma independente e realizam apenas uma única etapa de comunicação com o servidor, reduzindo significativamente a sobrecarga de comunicação típica dos métodos iterativos tradicionais. Já técnicas de *knowledge distillation* [Qin et al. 2025, Zhu et al. 2021] em FL permitem transferir o conhecimento de múltiplos modelos locais (frequentemente maiores) para um modelo global menor ou mais eficiente, agregando informações através de logits, representações intermediárias ou pseudo-rótulos em vez de compartilhar diretamente os parâmetros completos dos modelos [Papernot et al. 2016]. Essas estratégias tornam possível reduzir tanto o custo de comunicação quanto o computacional nos dispositivos dos clientes, sendo particularmente relevantes para cenários com recursos limitados e para aplicações envolvendo modelos de linguagem.

3. Metodologia

Esta seção apresenta os componentes do FedDALE através da metodologia de ajuste eficiente dos SLMs utilizando transferência do conhecimento e FL de forma não-supervisionada.

3.1. Visão Geral

A visão geral do FedDALE e seus processos estão descritos na Figura 1 que ilustra todos os passos e componentes presentes na proposta, especificados pelo que deve ocorrer no servidor ou no cliente. Nesse caso, o cliente está representado por um dispositivo de baixo poder computacional como um *smartphone* porém o tipo de dispositivo pode variar a depender da aplicação na qual está sendo utilizado sem perda de generalidade dos seguintes passos.

O primeiro passo ① do FedDALE envolve o treinamento local, nos clientes, do modelo professor de maneira paralela e completamente não-supervisionada utilizando as técnicas de aprendizado auto-supervisionado (detalhes na Seção 3.2). Em seguida, em ②, os clientes

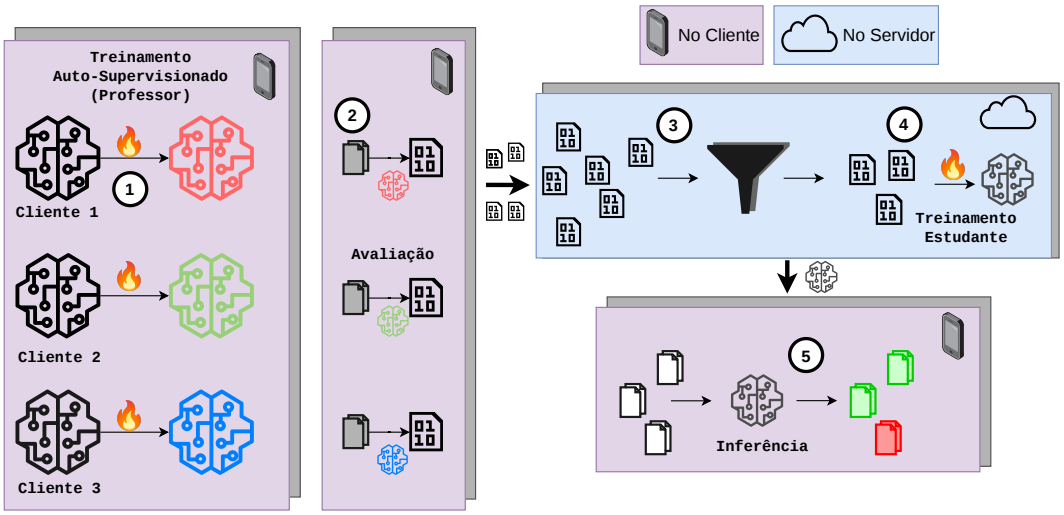

Figura 1. Ilustração do passo a passo de funcionamento do FedDALE entre o cliente treinando o modelo professor e o servidor preparando os dados e treinando o modelo estudante final a ser utilizado pelos clientes para detecção de anomalias.

utilizam um conjunto de dados de *logs* globais comum entre eles e sem rótulos, que podem advir de alguma aplicação ou conjunto de dados público sem perda de privacidade, para a coleta da performance dos modelos dos locais nesse conjunto de validação. Essas informações são processadas e transformadas em métricas de predições atribuídas pelos próprios modelos dos clientes seguindo protocolo detalhado na Seção 3.3. Estas métricas, ao invés dos modelos inteiros, são enviadas para o servidor e passam por ③ uma transformação em pseudo-rótulos e uma filtragem de qualidade que visa selecionar os melhores dados, e remover os mais incertos, para o treinamento do modelo estudante.

Então, em ④, os dados filtrados são utilizados para o treinamento supervisionado de classificação do modelo estudante, com os pseudo-rótulos atribuídos pelos professores. Dessa forma, o conhecimento acerca do comportamento padrão dos *logs* é extraído do professor para o estudante, que é muito menor e mais eficiente para detecção. Por fim, em ⑤, o modelo estudante é enviado para os clientes e pode ser utilizado para classificação de novos dados entre anômalos e normais. Todo esse processo pode ocorrer em uma única rodada ou pode ocorrer de forma iterativa por algumas rodadas de FL para no final obter o modelo estudante eficiente e performático (detalhes na Seção 3.4).

3.2. Treinamento do Modelo Professor

A primeira etapa do FedDALE se baseia na preparação e treinamento do modelo professor, que acontece do lado dos clientes. Esse modelo é o maior o qual será utilizado na aplicação final e tem por função aprender o comportamento dos dados dos clientes para que depois esse aprendizado seja transferido para um modelo menor e mais eficiente na detecção. O treinamento dos modelos professores inclui o pré-processamento dos dados de *logs*, o ajuste eficiente e completamente não-supervisionado.

Pré-Processamento dos Dados. A etapa de pré-processamento foi adotada seguindo as técnicas propostas na literatura de detecção de anomalias [Guo et al. 2021, Almodovar et al. 2024, Talasso et al. 2025, Zhu et al. 2023]. A primeira etapa desse processo é a remoção de informações que poderiam prejudicar o treinamento dos modelos, como IDs do sistema e horários, ou representar riscos à privacidade dos usuários, como endereços

IP e links. Isso é feito através de uma expressão regular para remoção dos números e padrões presentes nesses tipos de dados por uma sequência especial: "<*>". Essa substituição de valores permite que o modelo reconheça que naquela posição existiam números, porém evita o modelo da tentativa de aprender essas sequências evitando gerar confusões durante a predição, uma vez que o modelo não será penalizado na predição dos números. Adicionalmente, esse processo auxilia na privacidade evitando memorização do modelo uma vez que remove informações sensíveis como os IPs.

A segunda etapa do pré-processamento envolve a separação das mensagens. Para isso, utiliza-se uma sequência especial, *";-;"*, para delimitar cada mensagem dos *logs*. Assim, é possível que o modelo aprenda não apenas quais mensagens podem representar anomalias, mas também possa aprender quais sequências de diferentes mensagens representam um comportamento anômalo. Por fim, todos esses dados serão janelados na capacidade máxima do modelo a ser treinado e tokenizados para servirem de entrada para o treinamento.

Ajuste Eficiente. Como os LLMs, e até mesmo as versões mais eficientes como os SLMs contém geralmente centenas de milhões ou até bilhões de parâmetros, quando se tratado de dispositivos de baixo recurso outras técnicas são necessárias para aplicar o treinamento eficiente nesses modelos. Para mitigar esse problema, propomos o uso de uma técnica de *fine-tuning* eficiente em parâmetros (*Parameter-Efficient Fine-Tuning* – PEFT), denominada *Low-Rank Adaptation* (LoRA) [Hu et al. 2021]. O LoRA é amplamente utilizado para adaptar modelos de linguagem de grande escala de forma eficiente, pois mantém os pesos do modelo pré-treinado congelados e introduz apenas um pequeno conjunto de parâmetros treináveis. Dessa forma, reduzem-se significativamente o custo computacional e a quantidade de dados que precisam ser comunicados no processo federado.

A técnica consiste em inserir matrizes de baixo posto em determinadas camadas do modelo, permitindo adaptar o comportamento do modelo para novas tarefas sem modificar os pesos originais. Formalmente, os parâmetros ajustados durante o treinamento são representados como uma decomposição de baixo rank aplicada sobre os pesos congelados do modelo:

$$W = W_0 + w = W_0 + AB \tag{1}$$

onde $W_0 \in \mathbb{R}^{d\times k}$ representa a matriz de pesos original do modelo pré-treinado, enquanto $A \in \mathbb{R}^{d\times r}$ e $B \in \mathbb{R}^{r\times k}$ são matrizes treináveis de baixo rank, com $r \ll d$ e $r \ll k$. Essa decomposição permite representar a atualização de pesos w com um número muito menor de parâmetros, mantendo a dimensionalidade final da matriz atualizada $W \in \mathbb{R}^{d\times k}$.

No contexto federado, essas matrizes de adaptação funcionam como adaptadores específicos de cada cliente. Assim, apenas os parâmetros LoRA precisam ser transmitidos ao servidor central durante as rodadas de agregação, reduzindo significativamente o custo de comunicação. Além disso, como o número de parâmetros treináveis é pequeno, o custo computacional local também é reduzido. Dessa forma, o modelo consegue aproveitar o conhecimento prévio do LLM enquanto se adapta ao padrão normal de *logs* observados localmente, permitindo realizar a detecção eficiente de anomalias.

Treinamento Não-Supervisonado. Após a etapa de pré-processamento dos dados e a definição da técnica de ajuste eficiente, inicia-se a primeira etapa ① de treinamento do FedDALE. Essa etapa consiste no treinamento dos modelos locais de cada cliente, denominados modelos professores, e é feita de forma completamente não supervisionada. Cada cliente realiza o treinamento localmente e de maneira paralela, explorando apenas os *logs* disponíveis

em seu próprio ambiente, sem a necessidade de compartilhamento de dados brutos.

A utilização de modelos maiores nessa etapa permite aprender representações mais precisas e completas do comportamento normal dos dados, o que resulta em maior qualidade do conhecimento que será utilizado posteriormente no sistema para o treinamento do estudante. Além disso, modelos com maior capacidade tendem a capturar padrões relevantes em menos rodadas de treinamento, reduzindo o número de interações necessárias no processo federado e, consequentemente, os custos de comunicação e computação. O tamanho desses modelos deve, entretanto, respeitar as limitações de hardware dos dispositivos presentes na federação.

Durante esse processo, apenas os adaptadores LoRA [Hu et al. 2021] são treinados, mantendo congelados os pesos do modelo pré-treinado. Essa estratégia reduz significativamente a quantidade de parâmetros atualizados e, consequentemente, o volume de dados que precisará ser transmitido ao servidor nas etapas subsequentes. O treinamento é realizado utilizando a tarefa auto-supervisionada de predição da próxima palavra, que constitui uma tarefa auto-supervisionada, evitando a necessidade de rótulos *logs* [Almodovar et al. 2024].

3.3. Treinamento do Estudante

Após os modelos professores serem treinados de forma distribuída e privada em paralelo nos clientes, cada um deles é responsável por coletar as métricas associadas ao modelo locais e enviá-las ao servidor. O servidor então será responsável por realizar as próximas etapas do FedDALE , sendo elas a rotulagem de dados pela validação de performance, o filtro de qualidade e posteriormente o treinamento do modelo estudante final a ser utilizado para detecção.

Rotulagem de dados. O FedDALE performa, no passo ②, a coleta das métricas dos modelos que serão utilizadas na criação dos pseudo-rótulos de treinamento do modelo estudante. Nessa etapa, um conjunto de dados públicos, os quais podem ser extraídos de aplicações públicas e disponibilizados sem problemas de privacidade e sem a necessidade de rótulos, é utilizado para avaliação de modelos professores locais. Similar com o proposto em [Talasso et al. 2025, Almodovar et al. 2024] as acurácias nas predições de próximas palavras são extraídas para uma sequência de *logs* e utilizadas como métricas da qualidade desses modelos nas amostras. Se as acurácias são altas, significa que os modelos aprenderam bem o comportamento daquela amostra, indicando normalidade dos dados, já se os modelos performam mal na predição, significa um indício de anormalidade uma vez que o dado é diferente do que foi visto no treino [Guo et al. 2021, Li et al. 2022]. Em seguida, apenas essas métricas de acurácia, para cada amostra do conjunto público, são enviadas ao servidor, representando um custo muito menor que o envio dos parâmetros do modelo.

Filtragem dos dados. Em seguida, após a coleta das acurácias de cada uma das amostras, o servidor performa um filtro de qualidade dos dados em ③. Como as acurácias são geradas por modelos professores que cometem erros e ainda não foram inteiramente treinados, confiar puramente nas performances atribuídas pode incorporar ao treinamento dados de pior qualidade, prejudicando o modelo estudante. Dessa forma, nessa etapa o FedDALE aplica um filtro de qualidade pegando um subconjunto dos dados totais mais confiáveis.

Primeiramente, a filtragem é feita através da agregação das predições dos diferentes professores através da média de acurácia atribuída por cada um deles no conjunto de dados. Em seguida, essas métricas agregadas são então separadas em conjuntos com maior confiança nas respostas, ou seja, são escolhidas as amostras que apresentaram maior acurácia média entre os clientes e aquelas que exibiram menor acurácia média. As que apresentaram acurácia mais alta são mais prováveis de serem normais, uma vez que diversos clientes aprenderam

a representar bem aqueles dados e as que tiveram menor acurácia são mais prováveis de serem anômalas, uma vez que a maioria dos clientes falharam em representá-las. Por fim, os pseudo-rótulos binários estão aplicados nesses subconjuntos de maior e menor acurácia média.

O tamanho do conjunto de confiança é um hiperparâmetro do FedDALE e foi fixado nos nossos testes em 400 amostras no total (200 maiores acurácias e 200 menores), uma vez que pegar conjuntos muito grandes acaba adicionando dados mais incertos para o treinamento e conjuntos muito pequenos podem ser insuficientes para treinar o modelo estudante. Adicionalmente, outros métodos de filtragem foram propostos e testados, como por exemplo utilizando a acurácia máxima ao invés da média na agregação, e os resultados dessa variação são mostrados com detalhes na Seção 4.3 de resultados.

Treinamento Supervisionado do Estudante. Com os dados rotulados de maneira binária, anomalias e normais e filtrados para terem mais qualidade, a próxima etapa ④ representa o treinamento supervisionado do modelo estudante, também utilizando ajuste eficiente com LoRA. Nessa etapa um modelo muito menor que o modelo professor é treinado para classificação nos pseudo-rótulos atribuídos, transferindo assim o conhecimento do modelo maior para um modelo menor e mais eficiente que posteriormente é enviado para os clientes novamente detectarem anomalias em ⑤.

3.4. Aprendizado Federado

As etapas anteriores de treinamento dos modelos professores e estudantes se referem a apenas uma rodada de comunicação cliente-servidor, conhecida na literatura como *one-shot*, ou uma tentativa. Porém, como o modelo estudante é treinado diretamente nos dados rotulados pelo modelo professor, dessa forma tendo sua performance completamente condicionada ao treinamento local, permitimos que o FedDALE performe algumas rodas de FL tradicional visando melhorar os modelos professores, abordagem conhecida como *few-shot*, ou poucas tentativas, e como consequência treinar melhores modelos estudantes posteriormente.

Seguimos nesse caso a abordagem clássica do FL, o *Federated Averaging* (FedAvg) [McMahan et al. 2017], porém, qualquer outro método do estado-da-arte de FL, seja para comunicação eficiente [Souza et al. 2023], melhores agregações de modelos [Li et al. 2020b, De Oliveira Jarczewski et al. 2026] ou até mesmo a criação de modelos personalizados [Kulkarni et al. 2020, Talasso et al. 2024, Talasso et al. 2026] pode ser aplicado, uma vez que qualquer melhora nos modelos professores dos clientes impacta na performance dos modelos estudantes positivamente sem perda de generalidade da proposta.

Dessa forma, a Figura 3.4 ilustra o processo caso seja necessário o treinamento por mais rodadas. Igualmente, todos os clientes começam treinando os modelos professores localmente. Em seguida, os modelos são comunicados ao servidor que deve tomar uma decisão de (i) agregar os modelos seguindo o FedAvg e reenviar aos clientes para um novo treinamento local ou (ii) seguir os passos descritos anteriormente (na Seção 3.3) e finalizar o treinamento do modelo estudante. Essa decisão pode ser baseada em critérios de convergência dos modelos professores, performance necessária do estudante, número de rodadas pré-definidas ou até mesmo baseada em requisitos e limitações de recursos de computação e comunicação dos clientes. O impacto das duas opções é mostrado em detalhes na Seção 4.

4. Análise de Desempenho

Esta seção detalha a implementação dos experimentos no FedDALE. Para a análise de custo e performance, consideramos modelos pequenos de linguagem (SLMs).

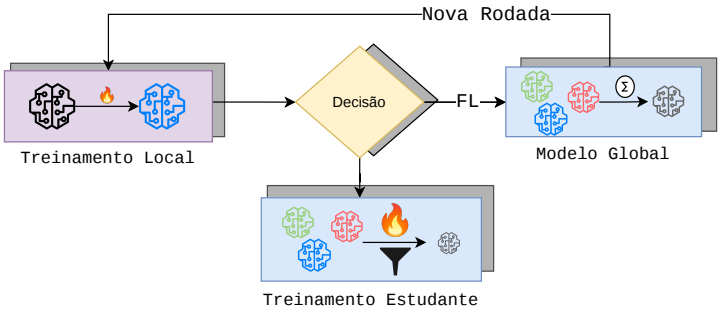

Figura 2. Ilustração do processo do FedDALE no caso de um aprendizado federado de *one-shot* ou de *few-shot* para o treinamento dos modelos professores.

4.1. Cenário de Avaliação

A família de modelos escolhida foi a SmolLMs [Allal et al. 2024], composta por versões com parâmetros variando entre 135 milhões e 1,7 bilhões. No caso do FedDALE foram utilizados como modelos professores os maiores 1.7B, enquanto os estudantes foram fixados nos modelos de 135M de parâmetros, combinando assim performance dos modelos maiores com a eficiência dos menores.

Os treinamentos utilizaram técnicas de ajuste eficiente com LoRA, mantendo o parâmetro de rank em 8 [Talasso et al. 2025]. Além disso, todos os modelos foram treinados e armazenados utilizando quantização para 4 bits, reduzindo a memória necessária durante o treinamento. Os experimentos foram realizados em um cluster de GPUs NVIDIA A100, cada uma com 80 GB de memória, quando não especificado, outros *hardwares*.

O conjunto de dados utilizado para experimentação foi Hadoop Distributed File System (HDFS) [Xu et al. 2009], que contém *logs* normais e anômalos. Cada cliente recebe uma parcela igual dos 400 mil dados de treinamento criados após o pré-processamento. Para teste foram considerados um conjunto de 1000 amostras normais e 1000 anômalas assim como no conjunto público de validação para coleta de métricas. Todos os conjuntos foram criados de forma sem interseção para melhor avaliação. A métrica utilizada para a qualidade da detecção foi F1-score, que balanceia capacidade de detecção com precisão ao acusar anomalias.

Todos os experimentos foram conduzidos com 15 rodadas de comunicação no FL padrão e até 5 rodadas para o *few-shot* do FedDALE em uma federação de 10 clientes com dados IID (Independentes e Identicamente Distribuídos). Além disso, para o treinamento dos modelos professores foi utilizada uma taxa de aprendizado com decaimento cossenoidal [Ye et al. 2024], variando de 10^{-3} até 10^{-5}, com um tamanho de *batch* de 16 e um total de 10 interações por rodada por cliente e para os estudantes, no servidor, uma taxa de 2×10^{-4} em *batches* de 64 por 100 iterações. Os resultados mostram as médias de 5 execuções de cada experimento com diferentes *seeds*. A implementação está publicamente disponível[2].

4.2. Resultados

Performance. Para avaliar a performance do FedDALE quanto a sua capacidade de detecção de anomalias, o primeiro experimento realizado demonstra sua performance, em diferentes rodadas de comunicação (1, 3, 5) quando comparado com um aprendizado federado comum (seguindo FedAvg [McMahan et al. 2017]) com os modelos de 135 milhões e de 1.7 bilhões

[2]Código disponível em: https://github.com/GabrielTalasso/FedDALE

de parâmetros e sem as técnicas de transferência de conhecimento e filtragem de dados propostas. A Figura 3 apresenta os resultados dessa comparação.

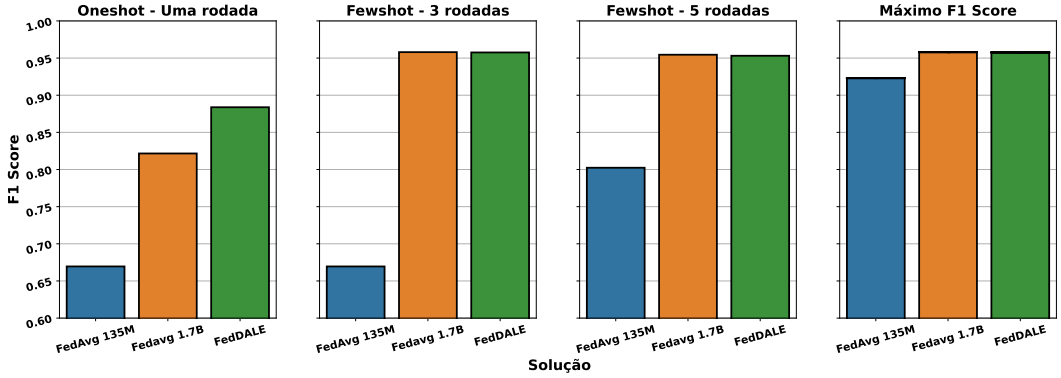

Figura 3. Performances do FedDALE quando comparado com FedAvg para diferentes rodadas de comunicação.

É possível observar, através dos resultados obtidos de performance em termos do F1-score das soluções, que, primeiramente, o FedDALE atinge uma performance mais alta na primeira rodada, demonstrando sua eficiência em cenários com pouca comunicação, uma vez que o FedDALE transmite apenas as predições e o FedAvg transmite os adaptadores inteiros dos modelos. Dessa forma, fica evidente a primeira vantagem do método proposto em cenários de comunicação muito restrita: o FedDALE supera as alternativas com apenas uma transmissão ao servidor central (*oneshot*) e sem a necessidade de mandar muitos parâmetros.

Além disso, nas rodadas seguintes (3 e 5), podemos observar que o FedDALE apresenta uma performance similar ao modelo de 1.7B de parâmetros mesmo contendo apenas 8% do tamanho desse modelo, demonstrando sua capacidade de manter alta performance na detecção de anomalias. Quando comparado com o modelo de mesmo tamanho final, como o 135M, o FedDALE atinge um F1-score até 20% maior na primeira rodada. Isso se deve ao fato de ele aprender, com um modelo muito menor, a imitar o comportamento do modelo professor, através da transferência de conhecimento e filtragem de qualidade propostas.

Comunicação. Para avaliar os custos de comunicação em cada uma das rodadas, a Tabela 1 mostra o tamanho das mensagens a serem transmitidas e recebidas por cada cliente em cada rodada. No caso do FedDALE, esse tamanho varia entre as rodadas de treino do professor e última rodada onde acontece o treino do estudante e apenas as métricas de performance são comunicadas e o modelo menor é recebido (Seção 3.3). Já na Figura 4 é apresentado o total gasto em comunicação em todas as rodadas com a soma de cada um clientes para o servidor. O total de comunicação é calculado pela soma dos valores das mensagens comunicadas por cada cliente e recebidas de volta do servidor (*uplink* e *dowlink*) em todas as rodadas.

É possível notar que o FedDALE comunica sempre menos que o modelo de 1.7B federado e mantém performance similar (como mostrado anteriormente) ou até o supera no caso do *oneshot* enquanto comunica 85.2% menos no total. Isso se deve ao fato de transmitir apenas as métricas do modelo estudante em vez de os adaptadores inteiros na última rodada, o que já é suficiente para apresentar uma redução significativa e vantajosa nos custos do detector de anomalias. Adicionalmente, o FedDALE comunica até 28.4% menos que ambas as soluções quando comparados na rodada de máximo F1-score, demonstrando sua capacidade de manter alta performance enquanto comunica de forma eficiente.

Detecção Eficiente. Para avaliarmos a eficiência do FedDALE quanto ao tempo para

Tabela 1. Comparação do tamanho das mensagens trocadas pelos clientes e servidor em MB.

Método	Uplink	Downlink
FL 135M	1.77	1.77
FL 1.7B	6.01	6.01
FedDALE (Prof.)	6.01	6.01
FedDALE (Estudante)	0.002	1.77

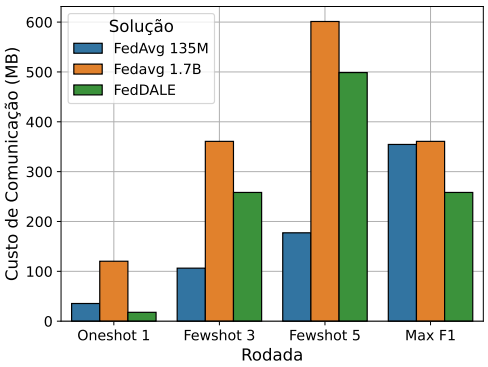

Figura 4. Comunicação total da soma de todos os clientes e rodadas. Max F1 representa a comunicação até a rodada de máxima performance.

a detecção de anomalias, isto é, o tempo de inferência dos modelos, testamos cada um dos modelos finais em diferentes hardwares no subconjunto de teste de 100 amostras. Os resultados, presentes na Tabela 2 mostram a média do tempo de inferência por amostra do conjunto e memória alocada para esse processo. Como o FedDALE cria ao final um modelo menor que o professor, sua capacidade de detecção é mais rápida, possibilitando sua aplicação em dispositivos mais restritos ou até mesmo para sistemas críticos que dependem de uma baixa latência ou detecção frequente.

Com a finalidade de deixar os testes mais realistas, utilizamos dispositivos reais para simular diferentes poderes computacionais, sendo eles: **Raspberry Pi5**, representando um dispositivo IoT de baixo poder computacional e sem GPU dedicada. A **NVIDIA Jetson Orin**, para representar um dispositivo de borda, embarcado, com um maior poder computacional e GPU incluída. Uma **NVIDIA RTX 4060 Ti** para demonstrar as capacidades em uma GPU mais acessível de uso pessoal ou menos intensivo. E por fim uma **NVIDIA A100**, que representa um grande servidor com alta capacidade computacional e uma das melhores GPUs disponíveis atualmente.

Tabela 2. Comparação da eficiência em diferentes *hardwares* reais.

Tempo de Inferência Médio (ms)					Uso de Memória Máximo (Mb)	
	Pi5	Jetson Orin	RTX 4060 Ti	A100	Modelo	Memória
FL 135M	65640.94	179.31	36.30	35.15	FL 135M	1007
FL 1.7B	120071.71	1492.80	174.40	119.83	FL 1.7B	7704
FedDALE	**55667.38**	**151.81**	**33.71**	**33.93**	**FedDALE**	**650**

Os resultados apresentados na Tabela 2 demonstram que o FedDALE mantém a eficiência de inferência quando comparado com modelos maiores, com uma redução de até 71% do tempo necessário para detecção de anomalias no caso da NVIDIA A100 e uma redução de mais de 7GB de memória de GPU. A diferença medida entre os modelos de mesmo tamanho no FL 135M e no FedDALE se deve ao fato do uso de KV-caching no modelo generativo que não é possível no classificador e a criação das representações da última camada, que no modelo generativo é muito maior que o modelo de classificação com apenas 2 saídas possíveis. Novamente, esse resultado representa que, mesmo mantendo a qualidade das detecções altas,

o FedDALE apresenta mais velocidade e eficiência, podendo ser aplicado em dispositivos mais restritos e aplicações mais críticas.

4.3. Ablação

Filtragem de Dados. Por fim, avaliamos o regime de filtragem de dados proposto no FedDALE, variando diferentes métricas para seleção dos melhores dados de treinamento do modelo estudante. Como proposto na Seção 3.3, o FedDALE utiliza as acurácias na predição de próximas palavras agregadas entre todos os clientes, dessa forma, a métrica utilizada para essa agregação foi escolhida entre (i) o máximo de acurácias entre os clientes para uma determinada amostra ou (ii) a média das acurácias para aquela amostra do conjunto público.

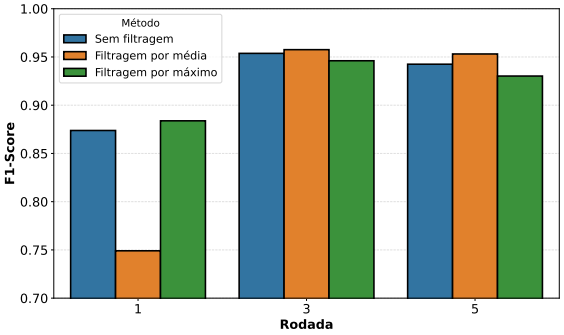

Figura 5. Diferentes regimes de filtragem de dados e seu comportamento ao longo das rodadas.

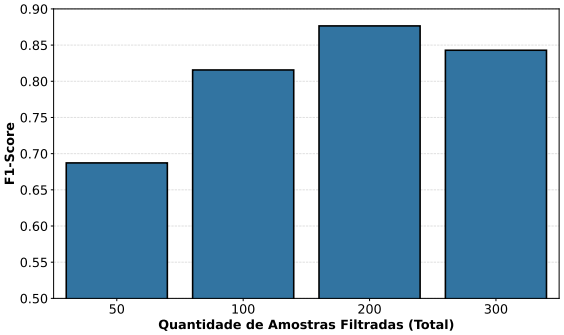

Figura 6. Performance sobre diferentes quantidades de dados selecionados para treinamento.

A Figura 5 apresenta a performance dessas abordagens em conjunto com a não-filtragem de dados, ou seja, sem agregar e filtrar as predições dos clientes. Os resultados demonstram primeiramente que a filtragem de dados melhora a performance do FedDALE em todas as etapas de treinamento. Porém, especialmente, a filtragem pelo máximo é melhor quando se trata de *oneshot* e pela média quando se trata das demais rodadas. Essa observação se deve ao fato de, nas primeiras rodadas, muitos modelos ainda não estarem treinados adequadamente, então utilizar o modelo que mais aprendeu (máxima acurácia) tem melhor separabilidade dos dados de anomalias e normais.

Por fim, como o FedDALE seleciona as amostras de melhor qualidade para o treinamento, a Figura 6 demonstra a performance, na primeira rodada utilizando a acurácia máxima, mas variando o número de amostras selecionadas. Os valores representam a quantidade pega com a menor acurácia agregada e com a maior acurácia agregada, resultando em um conjunto com duas vezes o tamanho mostrado. Os resultados demonstram uma troca entre ter amostras de maior qualidade e ter uma quantidade suficiente de dados de treinamento, onde poucas amostras muito boas são insuficientes e geram um *overfitting* e muitas amostras incorporam erros no treinamento, prejudicando o modelo.

5. Conclusão

Neste trabalho apresentamos o FedDALE, um método que une o aprendizado federado com transferência de conhecimento para criação de modelos performáticos de rápida detecção de anomalias enquanto reduz os custos associados à comunicação e ao uso dos modelos. Resultados em ambientes reais demonstram a superioridade de eficiência e performance contra outras soluções clássicas da literatura. Trabalhos futuros envolvem a expansão do FedDALE para mais domínios de *logs*, também envolvendo técnicas para mitigação de dados e recursos heterogêneos entre os clientes.

Disponibilidade de Artefatos

Em linha com princípios e práticas de Ciência Aberta, o conjunto de dados utilizado (Hadoop Distributed File System - HDSF [Xu et al. 2009]) pode ser realizado através do link: `https://github.com/logpai/loghub/tree/master/HDFS`. Da mesma forma, disponibilizamos a implementação do FedDALE através do repositório aberto acessível em: `https://github.com/GabrielTalasso/FedDALE`.

Agradecimentos

Este projeto foi apoiado pelo Ministério da Ciência, Tecnologia e Inovações, com recursos da Lei nº 8.248, de 23 de outubro de 1991, no âmbito do PPI-SOFTEX, coordenado pela Softex e publicado Arquitetura Cognitiva (Fase 3), DOU 01245.003479/2024-10.

Referências

Allal, L. B., Lozhkov, A., Bakouch, E., von Werra, L., and Wolf, T. (2024). Smollm - blazingly fast and remarkably powerful.

Almodovar, C., Sabrina, F., Karimi, S., and Azad, S. (2024). Logfit: Log anomaly detection using fine-tuned language models. *IEEE Transactions on Network and Service Management*.

De Oliveira Jarczewski, R., Cerqueira, E., Bittencourt, L. F., A. F. Loureiro, A., A. Villas, L., and De Souza, A. M. (2026). Participation is power: Effective approach to dynamic federated learning. In *Proceedings of the 18th IEEE/ACM International Conference on Utility and Cloud Computing*, UCC '25, New York, NY, USA. Association for Computing Machinery.

Devlin, J. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Guan, W., Cao, J., Qian, S., and Gao, J. (2024). Logllm: Log-based anomaly detection using large language models. *arXiv preprint arXiv:2411.08561*.

Guha, N., Talwalkar, A., and Smith, V. (2019). One-shot federated learning. *arXiv preprint arXiv:1902.11175*.

Guo, H., Yuan, S., and Wu, X. (2021). Logbert: Log anomaly detection via bert. In *2021 international joint conference on neural networks (IJCNN)*, pages 1–8. IEEE.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., and Chen, W. (2021). Lora: Low-rank adaptation of large language models. *CoRR*, abs/2106.09685.

Huong, T. T., Bac, T. P., Ha, K. N., Hoang, N. V., Hoang, N. X., Hung, N. T., and Tran, K. P. (2022). Federated learning-based explainable anomaly detection for industrial control systems. *IEEE Access*, 10:53854–53872.

Kulkarni, V., Kulkarni, M., and Pant, A. (2020). Survey of personalization techniques for federated learning. In *2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*, pages 794–797.

Li, B., Ma, S., Deng, R., Choo, K.-K. R., and Yang, J. (2022). Federated anomaly detection on system logs for the internet of things: A customizable and communication-efficient approach. *IEEE Transactions on Network and Service Management*, 19(2):1705–1716.

Li, Q., He, B., and Song, D. (2020a). Practical one-shot federated learning for cross-silo setting. *arXiv preprint arXiv:2010.01017*.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. (2020b). Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450.

Liu, Y. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 364.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. Pmlr.

Pang, G., Shen, C., Cao, L., and Hengel, A. V. D. (2021). Deep learning for anomaly detection: A review. *ACM computing surveys (CSUR)*, 54(2):1–38.

Papernot, N., Abadi, M., Erlingsson, U., Goodfellow, I., and Talwar, K. (2016). Semi-supervised knowledge transfer for deep learning from private training data. *arXiv preprint arXiv:1610.05755*.

Qi, J., Huang, S., Luan, Z., Yang, S., Fung, C., Yang, H., Qian, D., Shang, J., Xiao, Z., and Wu, Z. (2023). Loggpt: Exploring chatgpt for log-based anomaly detection. In *2023 IEEE International Conference on High Performance Computing & Communications, Data Science & Systems, Smart City & Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*, pages 273–280. IEEE.

Qin, L., Zhu, T., Zhou, W., and Yu, P. S. (2025). Knowledge distillation in federated learning: A survey on long lasting challenges and new solutions. *International Journal of Intelligent Systems*, 2025(1):7406934.

Souza, A., Bittencourt, L., Cerqueira, E., Loureiro, A., and Villas, L. (2023). Dispositivos, eu escolho vocês: Seleção de clientes adaptativa para comunicação eficiente em aprendizado federado. In *Anais do XLI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 1–14, Porto Alegre, RS, Brasil. SBC.

Talasso, G., de Souza, A., Guidoni, D., Cerqueira, E., and Villas, L. (2025). Fine-tuning eficiente de modelos de linguagem para detectar anomalias em logs privados usando aprendizado federado. In *Anais do XLIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 126–139, Porto Alegre, RS, Brasil. SBC.

Talasso, G. U., de Souza, A. M., Bittencourt, L. F., Cerqueira, E., Loureiro, A. A. F., and Villas, L. A. (2024). Fedsccs: Hierarchical clustering with multiple models for federated learning. In *ICC 2024 - IEEE International Conference on Communications*, pages 3280–3285.

Talasso, G. U., Kurmanji, M., de Souza, A. M., Lane, N. D., and Villas, L. A. (2026). Task-centric personalized federated fine-tuning of language models. *arXiv preprint arXiv:2604.00050*.

Wang, F., Zhang, Z., Zhang, X., Wu, Z., Mo, T., Lu, Q., Wang, W., Li, R., Xu, J., Tang, X., et al. (2024). A comprehensive survey of small language models in the era of large language models: Techniques, enhancements, applications, collaboration with llms, and trustworthiness. *arXiv preprint arXiv:2411.03350*.

Xu, W., Huang, L., Fox, A., Patterson, D., and Jordan, M. I. (2009). Detecting large-scale system problems by mining console logs. In *Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles*, SOSP '09, page 117–132, New York, NY, USA. Association for Computing Machinery.

Ye, R., Wang, W., Chai, J., Li, D., Li, Z., Xu, Y., Du, Y., Wang, Y., and Chen, S. (2024). Openfedllm: Training large language models on decentralized private data via federated learning.

Zhang, T., Gao, L., He, C., Zhang, M., Krishnamachari, B., and Avestimehr, A. S. (2022). Federated learning for the internet of things: Applications, challenges, and opportunities. *IEEE Internet of Things Magazine*, 5(1):24–29.

Zhu, J., He, S., He, P., Liu, J., and Lyu, M. R. (2023). Loghub: A large collection of system log datasets for ai-driven log analytics. In *2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)*, pages 355–366. IEEE.

Zhu, Z., Hong, J., and Zhou, J. (2021). Data-free knowledge distillation for heterogeneous federated learning. In Meila, M. and Zhang, T., editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12878–12889. PMLR.