Capítulo 2

Fine-tuning Federado de Modelos de Linguagem na Era da Comunicação

Allan M. de Souza, Joahannes B. D. da Costa, Daniel Guidoni, Gabriel Talasso, Filipe Maciel, Luis F. G. Gonzalez, Eduardo Cerqueira, Luiz F. Bittencourt, Antonio A. F. Loureiro, Leandro A. Villas

Abstract

This short course presents the fundamentals and advances in federated fine-tuning of large-scale language models (LLMs), an approach that combines model specialization with privacy preservation and data decentralization. Federated learning (FL) allows LLMs to be trained directly on the devices where the data is generated, avoiding their centralization. Parameter Efficient Fine-Tuning (PEFT) techniques, such as LoRA and QLoRA, make this process feasible on resource-constrained devices, reducing computational cost and the volume of transmitted data. Applications in areas such as healthcare, finance, software development, smart grids, and edge computing are explored. The main technical challenges, including data and device heterogeneity, memory and communication limitations, and research opportunities are also discussed. The short course includes a hands-on with the Flower framework, demonstrating federated fine-tuning of LLMs in practice. The content seeks to empower participants to develop ethical, scalable and efficient solutions with personalized and distributed AI.

Resumo

Este minicurso apresenta os fundamentos e avanços do fine-tuning federado de modelos de linguagem de grande escala (LLMs), uma abordagem que combina especialização de modelos com preservação da privacidade e descentralização dos dados. O aprendizado federado (FL) permite o treinamento de LLMs diretamente nos dispositivos onde os dados são gerados, evitando sua centralização. Técnicas de Parameter Efficient Fine-Tuning (PEFT), como LoRA e QLoRA, tornam esse processo viável em dispositivos com recursos limitados, reduzindo o custo computacional e o volume de dados transmitidos. São exploradas aplicações em áreas como saúde, finanças, desenvolvimento de software,

redes inteligentes e computação na borda. Também são discutidos os principais desafios técnicos, incluindo heterogeneidade de dados e dispositivos, limitações de memória e comunicação e também oportunidades de pesquisa. O minicurso inclui um hands-on com o framework Flower, demonstrando na prática o ajuste federado de LLMs. O conteúdo busca capacitar os participantes para desenvolver soluções éticas, escaláveis e eficientes com IA personalizada e distribuída.

2.1. Introdução

Nos últimos anos, testemunhamos uma revolução impulsionada pelos Modelos de Linguagem de Grande Escala (*Large Language Models (LLMs)*, na sigla em inglês), como o GPT, LLaMA, Gemini, Mistral e Claude. Esses modelos se destacam por sua capacidade de compreender e gerar texto com fluência, coerência e contextualização, sendo aplicáveis em uma infinidade de domínios, abrangendo áreas como processamento de linguagem natural, saúde, redes de comunicação e cidades inteligentes [Van Nguyen et al. 2024]. No entanto, seu uso eficaz requer um processo de especialização conhecido como *fine-tuning*, a qual realiza uma adaptação dos modelos a domínios específicos para que possam oferecer respostas mais precisas, seguras e contextualizadas.

Essa adaptação, no entanto é um processo custoso e complexo, pois o *fine-tuning* tradicional exige acesso a grandes volumes de dados de alta qualidade, frequentemente sensíveis, além de recursos computacionais consideráveis. Tais requisitos limitam sua aplicabilidade, especialmente em contextos onde a privacidade dos dados é uma prioridade, como na saúde e nas finanças, ou onde os recursos computacionais são escassos, como em dispositivos de borda ou infraestruturas descentralizadas. Isso cria uma barreira que impede que muitos dos benefícios dos LLMs sejam plenamente explorados [Hu et al. 2021].

Neste contexto, surge o Aprendizado Federado (AF) (*Federated Learning (FL)*, na sigla em inglês) como uma abordagem que permite o treinamento de modelos localmente nos dispositivos onde os dados estão armazenados [Capanema et al. 2025], o FL elimina a necessidade de centralizar dados sensíveis, reduzindo significativamente riscos de vazamento de informações e respeitando regulamentações como a LGPD (Lei Geral de Proteção de Dados) e o GDPR *(General Data Protection Regulation)* [McMahan et al. 2017]. Ao mesmo tempo, distribui o custo computacional entre os participantes da rede e permite a construção de modelos mais robustos, treinados em dados reais, heterogêneos e contextualizados. Mais do que uma alternativa ao paradigma centralizado, o FL representa permite ajustar um modelo genérico, permitindo que cada contexto (*i.e.*, cliente) contribua para a formação de um modelo global que respeite e reflita sua diversidade.

Ao combinarmos o *fine-tuning* com o FL, viabilizamos o *Fine-tuning Federado* de LLMs. Essa abordagem une a capacidade adaptativa dos LLMs e a arquitetura descentralizada do FL. O resultado é um modelo que aprende com diferentes fontes de dados sensíveis, sem comprometer sua segurança, e que se torna mais generalizável e robusto, graças à diversidade dos dados utilizados no treinamento. Entretanto, diversos desafios precisam ser endereçados uma vez que o compartilhamento de parâmetros de LLMs torna-se inviável devido a grande quantidade e muitas vezes condições de conexões não

confiáveis em cenários móveis.

Assim, soluções de *Parameter Efficient Fine-Tuning* (PEFT), como LoRA [Hu et al. 2021] e QLoRA [Dettmers et al. 2023], são essenciais para permitir o *fine-tuning* federado de LLMs reduzindo as barreiras de *hardware* para o *fine-tuning* e também reduzindo o *overhead* de comunicação. A ideia principal de abordagens de PEFT é ajustar uma pequena fração dos parâmetros do modelo, essas técnicas reduzem drasticamente o custo computacional e a sobrecarga de comunicação (i.e., dois gargalos críticos do FL). Assim, mesmo dispositivos com recursos limitados podem participar do processo de especialização de modelos, democratizando o acesso à inteligência artificial de última geração.

O *fine-tuning* federado é especialmente promissor em setores onde privacidade, personalização e eficiência são cruciais. Na saúde, por exemplo, permite treinar modelos capazes de auxiliar diagnósticos considerando particularidades regionais sem expor dados sensíveis de pacientes. No setor financeiro, pode ajudar na detecção de fraudes ou análise de crédito de forma contextualizada, respeitando legislações locais. No desenvolvimento de software, viabiliza assistentes de codificação que se adaptam ao estilo e práticas de uma equipe sem revelar código fonte. E em redes inteligentes, viabiliza modelos adaptativos para comunicação eficiente em ambientes heterogêneos, como na IoT ou em arquiteturas 6G.

Essas aplicações demonstram que o *fine-tuning* federado não é apenas uma alternativa técnica, mas uma solução estratégica para a criação de inteligência artificial ética, eficiente e inclusiva. É uma resposta concreta aos dilemas contemporâneos de centralização de dados, disparidades de infraestrutura e necessidade crescente de personalização nos serviços digitais. Por fim, vale destacar que o *fine-tuning* federado de LLMs está apenas começando a ser explorado. Ainda existem desafios importantes a serem superados, como a heterogeneidade dos dados e dispositivos, o balanceamento entre privacidade e desempenho, e a criação de novos algoritmos de agregação e coordenação federada. Mas são justamente esses desafios que tornam o campo fértil para inovação científica e tecnológica.

Este minicurso tem como objetivo não apenas apresentar os fundamentos técnicos e práticos do *fine-tuning* Federado de LLMs, mas também inspirar novas linhas de pesquisa, novos produtos e novas soluções para problemas reais. Ao dominar esses conceitos, pesquisadores, engenheiros e profissionais estarão aptos a participar ativamente da próxima onda da inteligência artificial: uma onda descentralizada, ética, personalizada e colaborativa.

O restante do documento está organizado da seguinte forma. A Seção 2.2 apresenta uma introdução aos modelos linguagens detalhando o processo de *tokenização*, *embeddings*, mecanismos de atenção, arquitetura dos *transformers*, pré-treinamento e as diferenças entre LLMs e SLMs. Em seguida Seção 2.3 descreve o processo de aprendizado federado completo e como clientes e servidor treinam o modelo de forma colaborativa. A Seção 2.4 apresenta o *fine-tuning* federado de LLMs introduzindo métodos de PEFT incluindo LoRA e QLoRA e como essas técnicas podem ser combinadas para permitir um *fine-tuning* eficiente de LLMs. A Seção 2.5 apresenta diversas aplicações onde o *fine-tuning* federado de LLMs pode ser uma tecnologia chave. A Seção 2.6 apresenta o

hands-on[1] para *fine-tuning* de LLMs apresentado desde a configuração do ambiente até o processo de simulação. A Seção 2.7 descreve os desafios e oportunidades de pesquisa para *fine-tuning* federado de modelos. Por fim, a Seção 2.8 conclui o documento apresentado uma visão geral do que foi abordado, desafios e oportunidades de pesquisa.

2.2. Modelos de Linguagem

Esta seção introduz os conceitos técnicos básicos para o entendimento do funcionamento de modelos de linguagem modernos tanto a nível de seu treinamento e otimização quanto a nível de inferência, ou seja, quando esses modelos forem utilizados em aplicações.

Os conceitos aqui tratados incluirão *(i)* tokenização do texto para o processamento na entrada do modelo, *(ii)* representação semântica e espacial de palavras através de *word-embeddings*, *(iii)* principais mecanismos de atenção utilizados pelos modelos de linguagem modernos (tais como *self-attention*, *masked-self-attention* e *multihead-attention*), *(iv)* arquitetura *transformers* [Vaswani 2017] clássica e arquitetura *decoder-only* dos modelos generativos atuais e *(v)* ajuste desses modelos em grande conjuntos de dados por predição de próxima palavra e *loss* de entropia-cruzada. Por fim, uma diferenciação tanto a nível de treinamento quanto a nível de utilização dos LLMs e Small Language Models (SLMs) será apresentada [Zhao et al. 2023]. De modo geral, esta seção tem como objetivo familiarizar o leitor com o tema e permitir um melhor entendimento dos conceitos que serão tratados nas próximas seções.

2.2.1. Tokenização de Palavras

O processo de geração de palavras com os modelos de linguagem atuais, os LLMs, conta com diversos passos para otimização tanto da performance desses modelos quanto também para torná-los cada vez mais eficazes e eficientes. O primeiro passo para a criação de LLMs é conhecido como tokenização de palavras e é responsável pela transformação de texto em números que podem ser processados através de redes neurais.

A tokenização é formada basicamente por um vocabulário responsável por fazer a transformação de texto para índices pré definidos de sequências de caracteres ou palavras. Como ilustrado na Figura 2.1, que exemplifica o tokenizador do modelo GPT-4 [OpenAI 2024] na frase *Aprendendo sobre "tokenizadores"!*, podemos ver que uma palavra pode ser composta de diversos *tokens*, assim como sequências de caracteres especiais também pode ser representado por único índice. São esses índices retornados pelo tokenizador que são utilizados como entrada dos modelos de linguagem.

Para a definição desse chamado vocabulário, seria possível utilizar um *token* para representar desde um único caractere, ou uma única palavra, até expressões inteiras em sequências de palavras. Por um lado, representar cada caractere como um *token* pode ser vantajoso por existir menos *tokens* possíveis e permitir o modelo a ter uma maior flexibilidade na geração e entendimentos das frases, porém o processo de treinamento sequencial seria extremamente mais custo uma vez que existiram muito mais passos por exemplo de treino. Por outro lado, gerar apenas palavras inteiras ou frases inteiras criaria um numero possível de *tokens* muito alto, dificultando o aprendizado do modelo daquelas

[1] `https://github.com/AllanMSouza/MC2-SBRC2025`

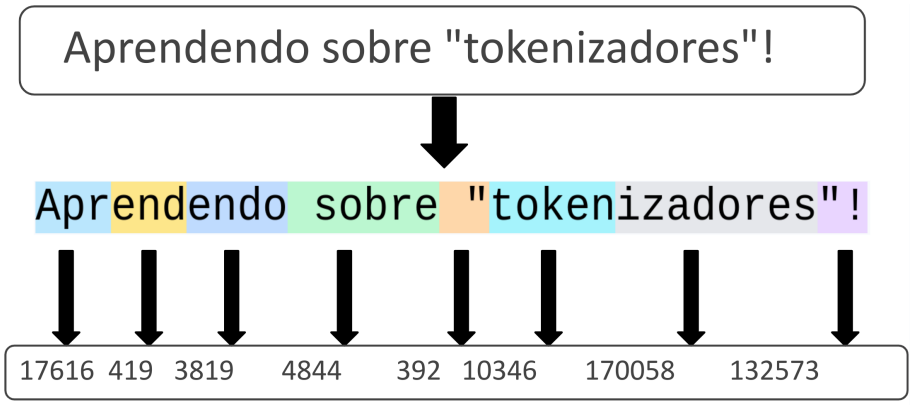

Figura 2.1. Ilustração do funcionamento do tokenizador do modelo GPT-4.

palavras em diversos contextos nas quais elas podem aparecer.

Dessa forma, na maioria dos modelos atuais, a tokenização ocorre em partes pré definidas de palavras, um meio termo entre as opções anteriores. Para determinar esses pedaços que serão definidos no vocabulário como *tokens*, é utilizado um processo chamado de *Byte-Pair Encoding* [Sennrich et al. 2016]. Esse processo se inicia com uma grande quantidade de textos de exemplos chamado de *corpus* e uma tokenização base a nível de caracteres. Em seguida, são examinadas as sequências de *tokens* mais frequentes nesse *corpus* utilizando essa tokenização inicial. Então, o par mais frequente, ou seja aqueles caracteres que apareceram juntos o maior número de vezes, serão transformados em um novo *token* que representa essa sequência de 2 caracteres. Esse processo é repetido, juntando sequências de *tokens* mais frequentes em um novo *token*, até que se atinja o tamanho do vocabulário pré-definido. Na maioria dos modelos esse tamanho gira em torno de 100 a 200 mil *tokens* possíveis [Llama-Team 2024].

Por fim, além de palavras encontradas no *corpus*, ainda são definidos nesse vocabulário alguns *tokens* chamados de "*tokens* especiais". Eles são definidos para desempenhar funções específicas como início e fim dos textos, tornando possível o modelo aprender quando parar de gerar palavras e terminar a frase, por exemplo. Porém, ainda é possível adicionar *tokens* especiais para ajustar o modelo em um pós treinamento, como por exemplo adicionar *tokens* de marcação de mensagens de chat, delimitando mensagens enviadas pelo usuário, mensagens de respostas anteriores do modelo e até mesmo regras de sistema. Esse tipo de marcação auxilia o modelo a entender melhor o contexto e se comportar da maneira mais adequada em cada caso.

2.2.2. Embeddings

Outro conceito bastante utilizado e muito importante no contexto de LLMs são os *embeddings*. Eles são vetores numéricos que representam não apenas palavras, mas carregam a semântica da palavra em um determinado contexto. Os *embeddings* são modificados ao longo do LLM, sendo operados pelas camadas de atenção de forma a adicionar sentido.

Um ponto importante desses vetores é baseado no fato de que, como eles carregam o significado da palavra entre múltiplas dimensões, é possível realizar operações numéricas nestes a fim de encontrar novas representações. Um exemplo é de que podemos

medir a similaridade de cossenos entre os *embeddings* de diferentes palavras e observar realmente se o significado delas são parecidos ou se são antagônicos, o que pode ser muito útil em sistemas de busca, por exemplo. Ainda, é possível fazer operações como utilizar o *embedding* da palavra "Paris", subtrair o da palavra "França"e adicionar o da palavra "Brasil"chegando a algo próximo do *embedding* de "Brasília".

Os primeiros modelos de geração de *embeddings*, como por exemplo o *Word2Vec* [Mikolov et al. 2013], são treinados para representar palavras em um contexto de forma estática, ou seja, a mesma palavra vai sempre gerar o mesmo vetor numérico. Especificamente esses modelos utilizam redes neurais para treinar a representação das palavras em muitos contextos diferentes, gerando uma representação única que carrega o significado da palavra em média desses contextos. Essa abordagem resulta em problemas quanto a palavras que podem mudar de significado em diferentes contexto, como a palavra "laranja"que pode representar uma fruta ou uma cor a depender de onde está sendo utilizada.

Por outro lado, os modelos mais recentes de linguagem, como os LLMs, utilizam *embeddings* dinâmicos, ou seja, que mudam conforme contexto nos quais eles estão inseridos. Especificamente os LLMs são responsáveis por processar e alterar as representações das palavras dentro de suas camadas, dando mais "atenção"ou "importância"a certas palavras e certos significados ao longo do processamento. Esse tipo de abordagem auxilia no entendimento do contexto usado por esses modelos, além de tornar possível representações mais complexas e significativas de cada palavra.

2.2.3. Mecanismos de Atenção

O conceito mais importante e mais utilizado nos modelos de linguagem atuais são os mecanismos de atenção. Basicamente eles são operações com os *embeddings* que visam atribuir significados adicionais com base no contexto em que a palavra está inserida. Esses mecanismos são definidos por operações específicas com parâmetros treináveis, ou seja, os modelos são ensinados a dar atenção da maneira correta aos *tokens* corretos ao longo do treinamento.

Existem diversos tipos mecanismos de atenção sendo desenvolvidos nos últimos anos para os mais diferentes fins, porém estaremos focados no mecanismo mais utilizados em LLMs a "auto-atenção", ou *self-attention*, e suas variações que tornam possíveis o treinamento em escala de modelos preditores de palavras.

Self-Attention. Este mecanismo possui esse nome porque realiza operações na frase contra ela mesma, ou seja, calculando a atenção de *tokens* numa frase com os próprios *tokens* da frase. A operação de atenção definida em [Vaswani 2017], que será detalhada nas próximas seções, está representada na Equação 1 onde Q, K e V são matrizes de pesos a serem aprendidas no modelo *transformers* (2.2.4). Isto é, a operação recebe como entrada um conjunto de *embeddings*, formando uma matriz, que serão multiplicados pelos pesos Q, K e V separadamente. Após o processamento pela atenção, espera-se como resultado dessa operação *embeddings* modificados que representam também o peso de cada *token* da frase no *token* em questão.

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (1)$$

MultiHead-Attention. As múltiplas cabeças de atenção se referem ao processo de paralelismo associado ao processamento dos *embeddings* pela *Attention*. Basicamente, o processo mostrado na Equação 1 é separado em diversas matrizes Q^i, K^i e V^i onde i é o índice de cada uma das cabeças $i \in \{0,1,..,H\}$ que processara a mesma entrada em paralelo. Dessa forma, além de otimizar o treinamento por conta das operações independentes, esse mecanismo permite que cada uma das cabeças aprenda a representar significados diferentes e fazer outras transformações que também podem ser úteis no treinamento.

Masked-Self- Attention. A atenção com o uso de máscaras é bastante similar aos mecanismos anteriores e está liga ao *decoder* do modelo *transformers* que será explicado posteriormente (2.2.4). O mecanismo é basicamente o mesmo da *self-attention* mas ao invés de calcular a atenção de todos os *tokens* de um para um, a atenção de um *token* é calculada se referindo apenas ao *tokens* anteriores a ele na frase. Dessa forma, os cálculos são feitos sempre considerando o passado da frase, o que é especialmente útil no caso de modelos generativos como LLMs que não possuem acesso ao que ainda vai ser gerado, apenas ao que já foi fornecido no contexto.

2.2.4. Transformers e LLMs

Entendendo os conceitos apresentados anteriormente, podemos compreender melhor o funcionamento da arquitetura *transformers*, tão importante para a construção dos modelos generativos atuais. Essas redes surgiram com a ideia do uso da atenção em seus diferentes formatos para o processamento de texto em linguagem natural em um contexto de tradução. Na época, essas redes revolucionaram as aplicações de tradução por apresentar vantagens como o poder de paralelismo e facilidades quanto ao ajuste sobre o que antes era o estado da arte. Além disso, após os casos de uso iniciais, essa arquitetura se espalhou para os mais variados cenários, não se limitando apenas à linguagem, mas também sendo aplicados em imagens, séries temporais, áudio e muitos outros.

A Figura 2.2 apresenta uma ilustração dos principais componentes dessa arquitetura. Do lado esquerdo está representado o codificador da arquitetura (*encoder*) que é responsável por receber o texto de entrada e criar representações significativas em *embeddings*. Já do lado direito está o decodificador (*decoder*), que utiliza as representações do *encoder* mas também representações próprias das saídas anteriores no momento da predição. Esses blocos de *encoder-decoder* são repetidos diversas vezes para formar a arquitetura completa. Dentro deles temos os componentes apresentados na seção anterior de auto-atenção, além de camadas de redes neurais comuns chamadas de *Feed Forward* e operações de adição e normalização dos *embeddings* intermediários para mais estabilidade durante o treinamento.

Ainda é importante ressaltar que essa é a forma clássica desses modelos que foram muito adaptados para diversas situações posteriormente à sua publicação inicial. Um exemplo comum seria o uso apenas da parte do *encoder* visando a criação de *embeddings* significativos para serem utilizados em outras aplicações como classificação, identificação de entidades, entre outros. O mais famoso modelo desse tipo atualmente é

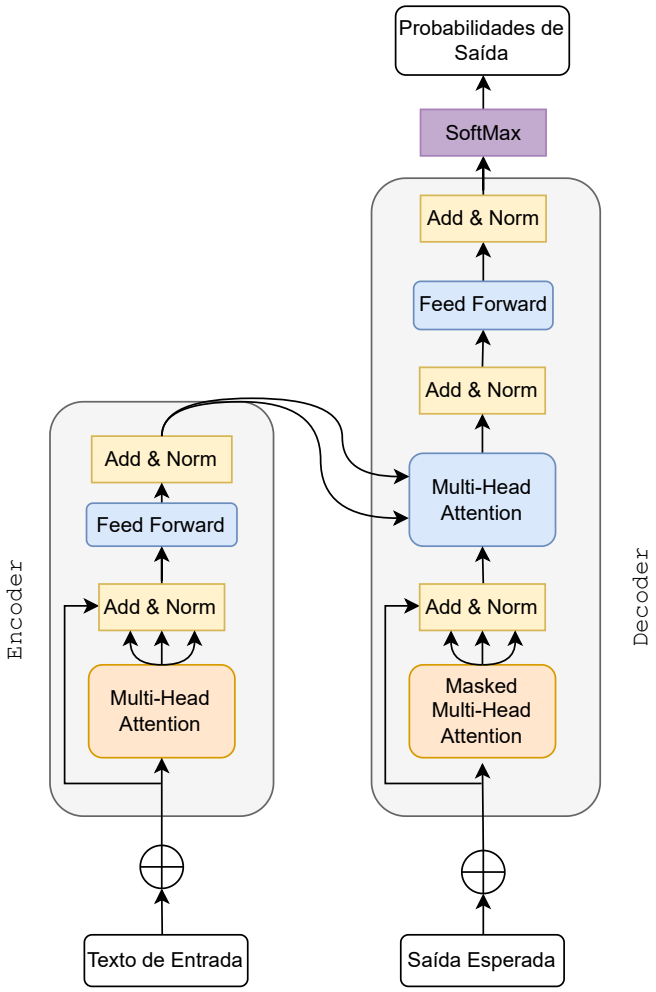

Figura 2.2. Ilustração do funcionamento da arquitetura *transformers* e seus componentes.

o BERT [Devlin et al. 2019], criado pela Google com a função de representar as palavras em uma arquitetura *encoder-only*.

Já no caso dos LLMs são utilizados apenas os blocos de *decoder* uma vez que eles são treinados para predizer palavras do próprio texto e utilizando a atenção mascarada para isso, levando em conta a penas as palavras anteriores a predição.

2.2.5. Pré-Treinamento

Pré-treinamento é o nome dado à fase de treino de LLMs em grandes bases de dados para adquirirem habilidades gerais sobre a linguagem. Essa é a fase mais custosa da criação desses modelos uma vez que é necessário blocasar todos os parâmetros, em uma enorme quantidade de dados por muitas iterações, consumindo quantidades significativas de memória, poder de processamento e tempo.

As grandes bases de dados, chamadas de *corpus*, podem ser criadas do zero ou podem ser utilizadas bases já existentes, separadas e filtradas, disponibilizadas a público [Penedo et al. 2024] e que muitas vezes são coletadas de grande parte pública da internet. Além disso, a qualidade na formatação e conteúdo nessa etapa é muito importante pois dados de baixa qualidade ou com informações indevidas e sensíveis pode prejudicar muito a performance e comportamento desses modelos.

Nessa etapa o treinamento é feito, no caso dos LLMs, com a tarefa única de predição de próxima palavra. Ou seja, a entrada é um texto retirado do *corpus*, passado pelo modelo para uma predição e depois ajustado para a palavra correta. A perda (*loss*) utilizada é bastante similar ao caso de classificação, mas a dimensão de saída do modelo são todos os *tokens* possíveis do vocabulário. Nesse caso utilização a entropia cruzada, como apresentado na Equação 2.

$$\mathscr{L}_{\text{CE}} = -\sum_{t=1}^{T} \log P(x_t \mid x_{<t}) \tag{2}$$

Onde, T é o tamanho da sequência e $P(x_t \mid x_{<t})$ é a predição da palavra em t dado o contexto anterior pelo modelo.

Dessa forma, a atribuição de probabilidade máxima para a palavra correta ($P(x_t \mid x_{<t}) = 1$) resulta em um acerto e consequentemente na menor *loss* possível. Assim, o modelo aprende a, dadas algumas palavras no contexto, atribuir mais probabilidade na próxima palavra. Esse processo se mostra eficaz pois ao aprender a predição de palavras o modelo internamente desenvolve diversas outras habilidade de compreensão e manipulação da linguagem, podendo depois ser utilizado em diversos contextos diferentes e até mesmo adaptado a novos cenários de maneira mais fácil, como veremos posteriormente [Radford et al. 2019].

2.2.6. LLMs e SLMs

Os Grandes modelos de Linguagem, com centenas de bilhões ou até mesmo trilhões de parâmetros, os quais, apesar de possuírem alta performance em diversas aplicações sendo utilizados para resolução de tarefas complexas, demandam alto poder de processamento e memória necessitando serem utilizados em grandes servidores com diversos hardwares especializados. Em contraponto, vêm surgindo soluções e desenvolvimento de modelos menores de linguagens, chamados de SLMs, os quais possuem alguns bilhões de parâmetros e são otimizados para demandarem menos recursos ao serem utilizados, possibilitando sua aplicação em contextos diferentes dos LLMs, como dispositivos de borda, dispositivos IoT, dispositivos móveis, ou até mesmo aplicações especializadas de baixa latência.

Os SLMs possuem um funcionamento e pré-treinamento bastante similar com o apresentado nas seções anteriores, visando seu aprendizado básico em diversos contextos. Porém, por possuírem um tamanho reduzido suas habilidades são limitadas muitas vezes não atingindo a performance dos LLMs em tarefas mais complexas e demandantes de "raciocínio". Dessa forma, geralmente utiliza-se os SLMs após uma especialização em dados da aplicação alvo, utilizando seu pré-treinamento genético como base de conhecimento para a melhora em tarefas específicas. Esse processo é chamado de fine-tuning

ou ajuste fino do modelo e permite que esses modelos mais eficientes performem em par com os modelos maiores em contextos específicos.

2.3. Aprendizado Federado

Nesta seção, apresentamos o FL destacando suas vantagens em relação a abordagem tradicional centralizada para treinamento de modelos. Dessa forma, apresentamos a ideia fundamental do FL, que endereça alguns pontos fracos que limitam o aprendizado de máquina clássico. Uma arquitetura de referência para um sistema de FL é apresentada. Com ela, os problemas associados à sua implementação e as soluções propostas, servindo como uma revisão literária do tema. Por fim, *frameworks* que possibilitam a utilização do FL em ambientes reais e de simulações. Ao final da seção o leitor deve se sentir familiarizado com os requisitos de aplicação do FL, os problemas relacionados à essa solução e quais ferramentas estão disponíveis para sua utilização.

2.3.1. Motivação

O aprendizado de máquina clássico [Faceli et al. 2021] utiliza um modelo e dados. Esse modelo pode ser uma rede neural, uma regressão linear ou outros. O objetivo é realizar uma tarefa útil com o modelo treinado com os dados. Na prática, os dados usados para treinar o modelo não vêm, necessariamente, da mesma máquina em que o treinamento acontece. Muitas vezes, esses dados são gerados em lugares distintos, como em *notebooks*, *smartphones* ou microcontroladores presentes em casas e carros inteligentes. Ou seja, há uma diversidade de dados originados de fontes diferentes que contribuem para a mesma tarefa.

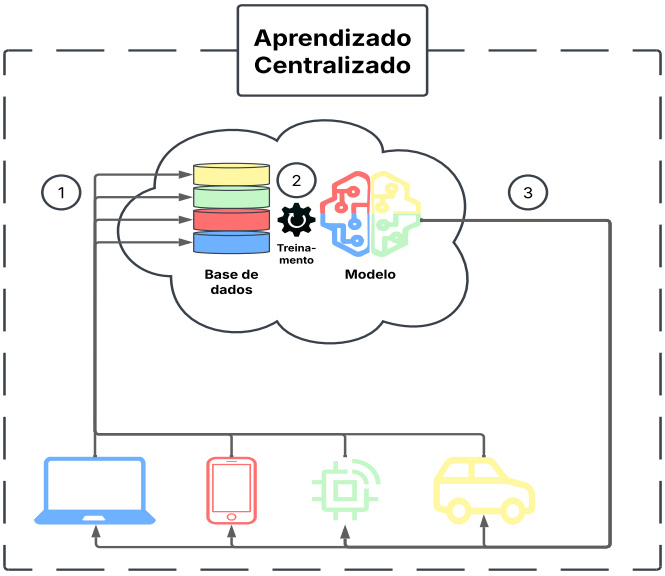

Figura 2.3. Ilustração do aprendizado de máquina clássico: 1 – Centralização dos dados, 2 – Treinamento do modelo e 3 – Disponibilização do modelo treinado.

Historicamente, o processo de um aprendizado de máquina consistia em coletar todos os dados em um servidor central, que poderia estar na nuvem. Assim que a reunião dos dados estivesse concluída, os algoritmos de aprendizado de máquina eram aplicados

para treinar o modelo, que então seria disponibilizado para a realização da tarefa. O processo é ilustrado na Figura 2.3.

O aprendizado de máquina centralizado é ideal quando todos os dados necessários para treinar um modelo estão disponíveis em um único lugar, como em um servidor central. Por exemplo, um servidor na web pode usar um modelo para prever o desempenho de um serviço e, com base no padrão de tráfego, ajustar seu funcionamento automaticamente [Filho et al. 2022]. Da mesma forma, uma plataforma de armazenamento de fotos na nuvem pode empregar modelos para agrupar imagens semelhantes e organizar capturas de tela e fotos de documentos em álbuns que sejam fáceis de localizar [Google 2023].

Existem várias situações em que o aprendizado de máquina tradicional não é a melhor opção. Por exemplo, há regulamentações que garantem que dados sensíveis dos usuários não sejam transferidos para um servidor central [da República 2018]. Além disso, muitos usuários preferem manter suas informações privadas [Wu et al. 2025a]. Outro ponto a considerar é o grande volume de dados em contextos onde os recursos são limitados, o que pode tornar os custos de comunicação muito altos [Lan et al. 2023].

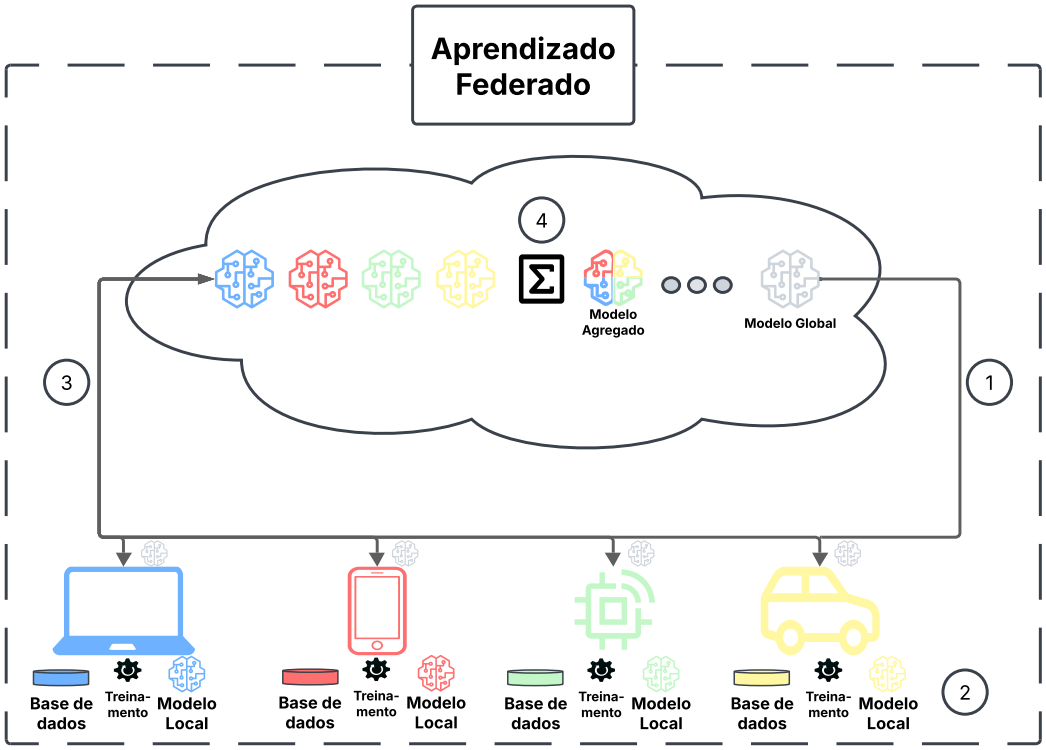

Figura 2.4. Ilustração do aprendizado de máquina federado: 1 – Envio do modelo global, 2 – Treinamento dos modelos locais, 3 – Envio dos modelos locais e 4 – Agregação dos modelos locais e repetição do processo.

A crescente popularidade de sistemas que priorizam a privacidade dos usuários, como o aplicativo de mensagens Signal [Signal 2013] e o navegador Brave [Brave 2015], mostra o quanto esse tema é relevante. É realmente vantajoso para os usuários utilizar um modelo que consiga detectar câncer sem precisar expor registros médicos durante o treinamento, identificar fraudes bancárias sem divulgar informações financeiras ou encontrar

a melhor opção de reabastecimento de veículos sem revelar a localização atual. Mas como podemos treinar esses modelos para fazer previsões sem comprometer a privacidade dos dados?

O FL muda a forma como o treinamento é feito em comparação com a abordagem tradicional, permitindo que o modelo seja treinado sem comprometer a privacidade dos usuários. Em vez de transferir os dados para treinar o modelo em um servidor central, ele leva o modelo até onde os dados estão, que são os próprios usuários. Então, os modelos locais são gerados a partir do treinamento do modelo global recebido do servidor. Esses modelos locais são retornados ao servidor central para agregação. O processo repete-se em rodadas de comunicação até a convergência do modelo ou um número pré-definido de rodadas. A Figura 2.4 ilustra essa ideia.

2.3.2. Arquitetura

A estrutura de um aplicativo que utiliza aprendizado federado está bastante ligada à configuração da rede que forma os recursos da federação [Wu et al. 2024b]. Normalmente, as aplicações mais frequentes adotam uma topologia centralizada, seja em forma de estrela ou hierárquica. Nessa configuração, há dois participantes principais: o servidor central e os clientes. O servidor central é responsável por coordenar o processo de aprendizado, enquanto os clientes realizam o treinamento com os dados que possuem localmente. Além desses participantes principais, existem outros componentes que oferecem funcionalidades básicas ao aplicativo. A arquitetura de referência [Lo et al. 2021] que ilustra esses elementos pode ser vista na Figura 2.5.

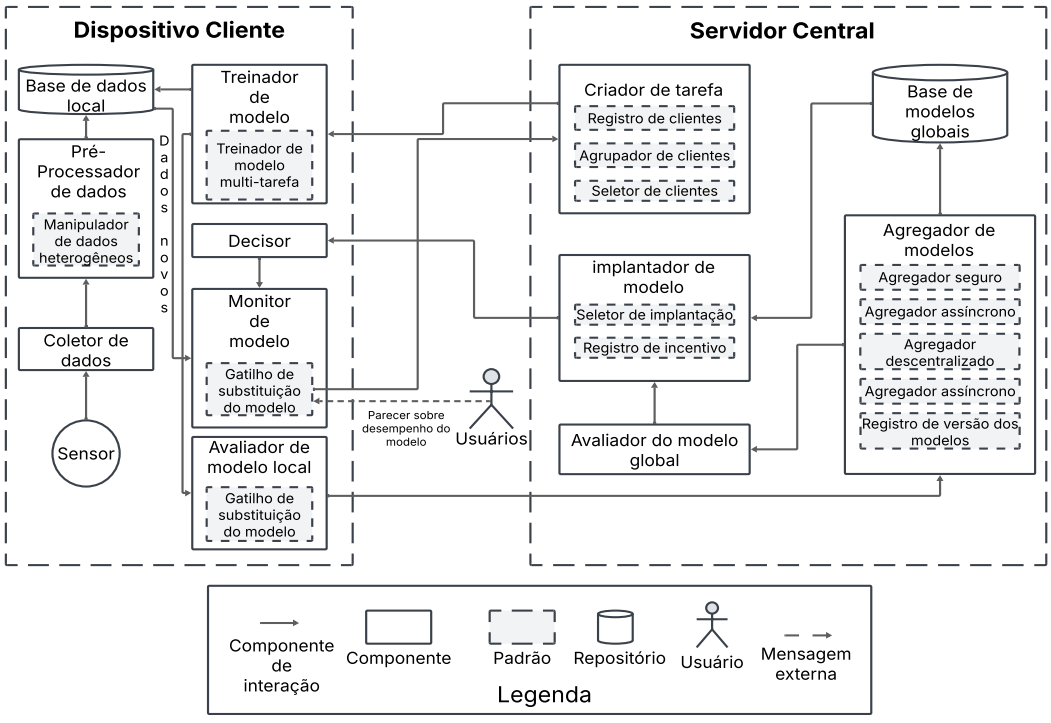

Figura 2.5. Arquitetura de referência obtida de [Lo et al. 2021].

O processo começa no servidor central, mais especificamente no componente chamado *Criador de tarefas*, onde é definido o modelo global que será treinado, além dos hiperparâmetros de treinamento. Alguns subcomponentes podem ser adicionados de forma opcional. O *Registro de clientes* armazena informações sobre os clientes, o que ajuda a otimizar, manter e aumentar a confiabilidade do sistema [Lo et al. 2022]. Já o *Agrupador de clientes* organiza os clientes com base em algum critério de similaridade, o que ajuda a minimizar os efeitos da diversidade no ambiente distribuído [Ye et al. 2023, Talasso et al. 2024]. Por fim, o *Seletor de clientes* orienta o aprendizado para que ele chegue a uma convergência mais rápida, escolhendo os melhores participantes para cada rodada de comunicação, considerando tanto a diversidade sistêmica quanto a estatística dos clientes [Fu et al. 2023, Souza et al. 2023, Maciel et al. 2024, de Souza et al. 2024, Jarczewski et al. 2024].

Cada dispositivo cliente possui um componente chamado *Coletor de dados*, que utiliza sensores para coletar os dados a serem processados pelo componente *Pré-processador de dados*. Este último pode incluir, de forma opcional, o subcomponente *Manipulador de dados heterogêneos*, que ajuda a reduzir o impacto da heterogeneidade estatística entre os clientes [Lu et al. 2024]. Quando o cliente recebe do servidor a tarefa de treinamento, ele utiliza os dados locais para treinar o modelo global. Em situações de treinamento multitarefa, é possível adicionar o subcomponente *Treinador de modelos multi-tarefa*, que permite o treinamento de modelos relacionados, aumentando assim o desempenho e a eficiência do aprendizado [Smith et al. 2017]. Após concluir o treinamento, o modelo local é avaliado pelo componente *Avaliador de modelo local* e enviado ao servidor para ser agregado posteriormente.

A agregação de modelos, feita pelo componente *Agregador de modelos*, é a tarefa de criar um novo modelo global a partir dos modelos que foram treinados localmente pelos clientes. Na literatura sobre aprendizado federado, existem várias propostas de agregadores [Nanayakkara et al. 2024]. Os subcomponentes do *Agregador de modelos* representam tipos comuns de agregadores que podem ser escolhidos para essa tarefa. Por exemplo, o subcomponente *Agregador seguro* se refere aos agregadores que acrescentam uma camada de verificação para garantir a autenticidade dos modelos treinados localmente, evitando que o novo modelo global seja alterado durante o processo de agregação. Por outro lado, o subcomponente *Agregador assíncrono* se refere aos agregadores que permitem que a agregação ocorra assim que o modelo local chega ao servidor, sem precisar esperar pela sincronia com os outros modelos locais. A classe de agregadores, que inclui o subcomponente chamado *Agregador descentralizado*, é focada em topologias descentralizadas. Já o subcomponente *Agregador hierárquico* é responsável por permitir camadas intermediárias de agregação em uma estrutura hierárquica. Por último, o subcomponente *Registro de versão dos modelos* pode ser adicionado ao agregador para aprimorar a governança e a rastreabilidade, facilitando a relação entre os modelos locais e o modelo global que eles geram.

Depois que a agregação é feita, a etapa de avaliação do desempenho do novo modelo global ocorre por meio do componente chamado *Avaliador de modelo global*. Essa avaliação serve como base para diversas soluções, como a seleção de clientes, o cálculo de contribuições, incentivos e clusterização, entre outras [Soltani et al. 2023]. Se o desempenho do modelo for considerado satisfatório, o componente *Implantador de modelo* envia

o novo modelo aos clientes, que têm a opção de decidir se irão utilizá-lo através do subcomponente *Decisor*. Os subcomponentes *Seletor de implantação* analisam quais clientes estão aptos a receber o novo modelo global com base em seus metadados e aplicações. Além disso, o componente *Registro de incentivo* mantém um registro sobre a conformidade dos clientes com o modelo global e quão grandes foram as contribuições que eles fizeram para sua evolução.

O monitoramento do modelo é feito de forma contínua nos clientes através do componente *Monitor de modelo*. Se o desempenho do modelo em algum cliente começar a piorar, o subcomponente chamado *Gatilho de substituição de modelo* envia uma mensagem para o servidor pedindo a criação de uma nova tarefa de treinamento.

Para reduzir o impacto da comunicação de modelos entre os clientes e o servidor, pode-se utilizar o componente *Compressor de mensagem*, que melhora a eficiência da comunicação ao diminuir a quantidade de bytes trocados. Isso é especialmente útil em contextos onde a largura de banda é limitada [Jia et al. 2025, Martins et al. 2024].

2.3.3. *Frameworks*

Existem vários *frameworks* de aprendizado federado disponíveis. *Riedel et al.* [Riedel et al. 2024] fizeram uma análise comparativa entre 15 deles, escolhidos com base em dois critérios: serem de código aberto e terem popularidade na comunidade. Eles observaram diversos aspectos qualitativos e quantitativos e atribuíram uma pontuação para cada uma das propostas. Entre as opções analisadas, o *framework* Flower [Beutel et al. 2020] se destacou com a melhor avaliação. Embora outros *frameworks* possam superar o Flower em algumas características específicas, nenhum deles se destacou em todas, o que justifica suas notas mais baixas na avaliação geral. Por exemplo, o FederatedScope [Xie et al. 2023] se destaca em recursos oferecidos ao usuário, enquanto o EasyFL [Zhuang et al. 2022] é mais fácil de usar. No entanto, o Flower também apresenta boas notas nessas áreas e se mostra superior em termos de interoperabilidade.

2.4. Fine-Tuning Federado de Modelos de Linguagem

O *fine-tuning* de modelos de linguagem permite adaptar modelos pré-treinados para domínios e tarefas específicos [Han et al. 2024], melhorando seu desempenho quando comparado com modelos sem tal adaptação. Entretanto, devido a grande quantidade de parâmetros desses modelos o seu ajuste pode ser custoso e necessitar muitos dados. Portanto, técnicas de *Parameter Efficient Fine-Tuning (PEFT)* foram propostas. A ideia principal é ajustar uma quantidade pequena de parâmetros do modelo, assim reduzindo o custo computacional e também a quantidade de dados necessárias para o ajuste.

Nesse cenário, a combinação de técnicas de PEFT com FL torna-se uma combinação poderosa, uma vez que podemos utilizar dados privados para o ajuste desses modelos além de aumentar a generalização do modelo devido a maior quantidade de dados utilizadas no *fine-tuning*. Além disso, o PEFT não só reduz a barreira de *hardware* necessária para ajustar os modelos, assim diversos dispositivos podem participar do processo de *Fine-tuning Federado*, mas também ataca desafios relacionados ao *overhead* de comunicação no *Fine-tuning Federado* uma vez que apenas os parâmetros ajustados precisam ser compartilhados com o servidor para agregação.

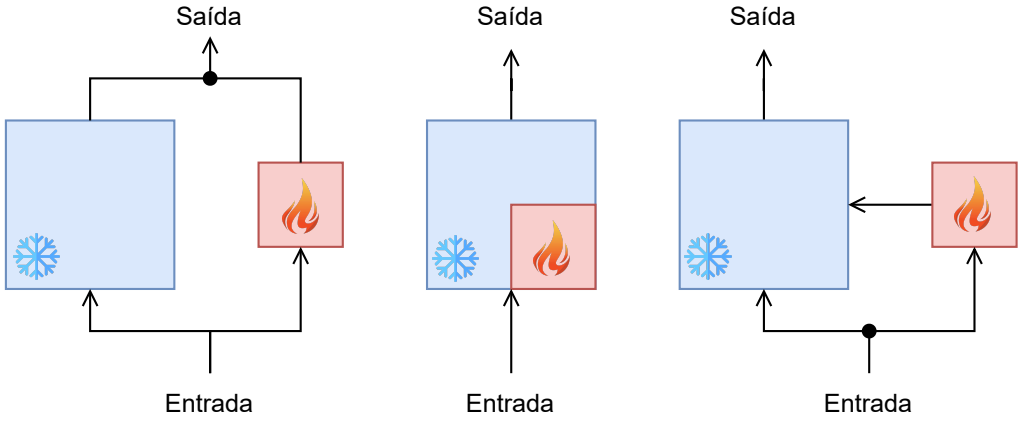

Figura 2.6. Diferentes estratégias de PEFT (a) PEFT aditivo, (b) PEFT seletivo e (c) PEFT reparametrizado

Esta seção apresenta uma breve introdução aos mecanismos para permitir um *fine-tuning* eficiente de modelos de linguagem. Assim, a Subseção 2.4.1 apresenta as diferentes abordagens para métodos de PEFT. A Subseção 2.4.2 descreve o LoRA, uma solução para *fine-tuning* eficiente de modelos, enquanto a Subseção 2.4.3 apresenta as otimizações introduzidas pelo QLoRA para reduzir a barreira de *hardware* para o *fine-tuning* de LLMs. Por fim, a Subseção 2.4.4 descreve como podemos combinar as tecnologias apresentados com o FL para permitir o *fine-tuning* federado de LLMs de forma eficiente e inclusiva.

2.4.1. Estratégias de PEFT

As estratégias PEFT podem ser amplamente classificadas em três categorias: *(i)* PEFT aditivo, que modifica a arquitetura do modelo injetando novos módulos ou parâmetros treináveis; *(ii)* PEFT seletivo, que torna um subconjunto de parâmetros treináveis durante o ajuste; e *(iii)* PEFT reparametrizado, que constrói uma reparametrização (de baixa dimensão) dos parâmetros do modelo original para treinamento. A Figura 2.6 apresenta as diferentes estratégias para PEFT, as quais serão descritas a seguir.

O *fine-tuning* completo padrão gera custos computacionais substanciais e também pode prejudicar a capacidade de generalização do modelo. Para mitigar esse problema, uma abordagem amplamente utilizada é manter o *backbone* pré-treinado inalterado e introduzir apenas um número mínimo de parâmetros treináveis, estrategicamente posicionados na arquitetura do modelo. Durante o ajuste fino para uma tarefa específica posterior, apenas os pesos desses módulos ou parâmetros adicionais são atualizados, o que resulta em uma redução substancial nos requisitos de armazenamento, memória e recursos computacionais [Hu et al. 2021].

É importante destacar que o PEFT aditivo aumenta a complexidade do modelo ao adicionar mais parâmetros, enquanto o PEFT seletivo ajusta um subconjunto dos parâmetros existentes para aprimorar o desempenho do modelo em tarefas subsequentes. Especificamente, dado um modelo com parâmetros $\theta = \{\theta_1, \theta_2, \ldots, \theta_n\}$ onde cada θ_i denota um parâmetro individual do modelo e n representa a contagem total desses parâmetros, o processo de PEFT seletivo é representado pela aplicação de uma máscara

binária $\mathscr{M} = \{m_1, m_2, \ldots, m_n\}$ a esses parâmetros. Cada m_i em $\mathscr{M}$ é 0 ou 1, indicando se o parâmetro correspondente θ_i está selecionado (1) ou não (0) para ajuste fino. O conjunto de parâmetros atualizado θ_1 após o ajuste fino é dado por:

$$\theta_i' = \theta_i - \eta \cdot m_i \cdot \frac{\partial \mathscr{L}}{\partial \theta_i} \tag{3}$$

onde η representa a taxa de aprendizado e $\frac{\partial \mathscr{L}}{\partial \theta_i}$ é o gradiente da função de perda em relação ao parâmetro θ_i. Nesta formulação, apenas os parâmetros selecionados (ou seja, $m_i = 1$) são atualizados durante o *backpropagation*. Algoritmos de PEFT seletivo são semelhantes aos métodos de *pruning* de modelos, definindo máscaras para remover neurônios ou conexões entre neurônios no processo de treinamento. Dessa forma, também são organizados como abordagens estruturadas (i.e., removendo neurônios) e não estruturadas (i.e., removendo conexões entre neurônios).

Reparametrização significa transformar equivalentemente a arquitetura de um modelo de uma para outra por meio da transformação de seus parâmetros. No contexto do PEFT, isso geralmente significa construir uma parametrização de baixo *rank* para atingir o objetivo de eficiência dos parâmetros durante o treinamento. Para inferência, o modelo pode ser convertido para sua parametrização de peso original, garantindo velocidade de inferência inalterada. Estudos de pesquisa anteriores [Aghajanyan et al. 2020] mostraram que modelos pré-treinados comuns exibem uma dimensionalidade intrínseca excepcionalmente baixa. Em outras palavras, é possível encontrar uma reparametrização de baixa dimensão que seja eficaz para o ajuste fino de todo o espaço de parâmetros. A técnica de reparametrização amplamente reconhecida é Low-Rank Adaptation (LoRA) [Hu et al. 2021] a qual será apresentada a seguir.

2.4.2. Low-Rank Adaptation - LoRA

O LoRA modifica apenas as matrizes de baixo *rank* associadas ao modelo, congelando a maioria dos parâmetros originais e introduzindo apenas ajustes leves. Isso reduz substancialmente o número de parâmetros atualizados durante o treinamento, tornando o *Fine-tuning Federado* mais eficiente em termos de comunicação e armazenamento. Além disso, o LoRA preserva o conhecimento adquirido pelo modelo base, evitando problemas como o esquecimento catastrófico.

Dessa forma, ao adaptar a uma tarefa específica, Aghajanyan e outros [Aghajanyan et al. 2020] mostram que os modelos de linguagem pré-treinados têm uma baixa "dimensão intrínseca"e ainda podem aprender eficientemente, apesar de uma projeção aleatória para um subespaço menor. Inspirados por isso, LoRA explora hipótese de que as atualizações dos pesos também têm uma baixa "classificação intrínseca"durante a adaptação. Para uma matriz de pesos pré-treinada $W_0 \in \mathbb{R}^{d \times k}$, restringimos sua atualização representando a última com uma decomposição de baixo *rank* $W_0 + \Delta W = W_0 + BA$, onde $B \in \mathbb{R}^{d \times r}$, $A \in \mathbb{R}^{r \times k}$ e o *rank* r $min(d,k)$. Durante o treinamento, W_0 é congelado e não recebe atualizações de gradiente, enquanto A e B contêm parâmetros treináveis. Observe que tanto W_0 quanto $\Delta W = BA$ são multiplicados pela mesma entrada, e seus respectivos vetores de saída são somados em coordenadas. A Figura 2.7 apresenta ilustra o processo de reparametrização aplicado pelo LoRA. Para $h = W_0 x$, assim o *feed*

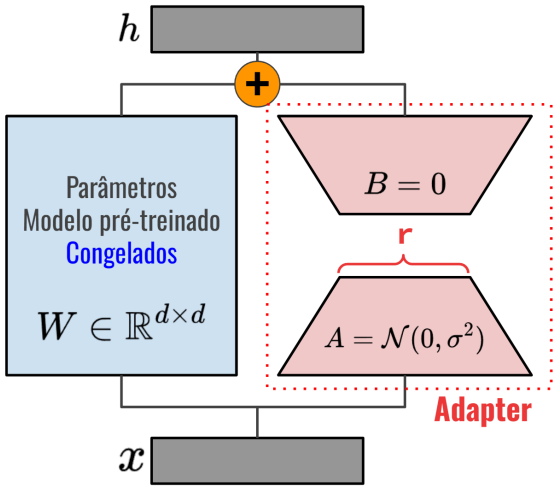

Figura 2.7. Reparametrização LoRA, onde apenas matrizes A e B são treinadas.

forward produz a seguinte saída:

$$h = W_0 x + \Delta W x = W_0 x + BAx \tag{4}$$

LoRA é Uma forma genérica de *fine-tuning* que permite o treinamento de um subconjunto dos parâmetros pré-treinados e não requer que a atualização de gradiente acumulada nas matrizes de ponderação tenha *rank* completo durante a adaptação. Isso significa que, ao aplicar LoRA a todas as matrizes de ponderação e treinar todos os vieses, recuperamos aproximadamente a expressividade do ajuste fino completo definindo o *rank* r para a classificação das matrizes de ponderação pré-treinadas. Em outras palavras, à medida que o número de parâmetros treináveis é aumentado, o treinamento de LoRA converge aproximadamente para o treinamento do modelo original, enquanto métodos baseados em adaptadores convergem para um MLP e métodos baseados em prefixos para um modelo que não pode receber sequências de entrada longas.

É importante destacar que o LoRA não introduz latência adicional no processo de inferência, pois é possível armazenar $W = W_0 + BA$ e realizar a inferência normalmente. Além disso, o mesmo modelo pode ser utilizado para diferentes tarefas, assim o conhecimento ajustado do modelo sempre será armazenados nos *adapters* (i.e., matrizes A e B). Dessa forma, caso for necessário utilizar o modelo para outra tarefa, basta apenas alterar o *adapter* treinado para a tarefa em questão, consequentemente introduzindo uma flexibilidade para o modelos e também reduzindo espaço de armazenamento, uma vez que apenas os novos *adapters* ajustados precisam ser armazenados e não um novo modelo ajustado.

2.4.3. QLoRA

Apesar do LoRA reduzir o custo computacional para o treinamento ele ainda necessidade que o dispositivo responsável pelo *fine-tuning* seja capaz de armazenar o modelo completo em memória. O que pode ser uma limitação para diversos dispositivos, assim novos métodos foram propostos como por exemplo o QLoRA [Dettmers et al. 2023], o qual introduz uma série de otimizações no LoRA para reduzir ainda mais a barreira de

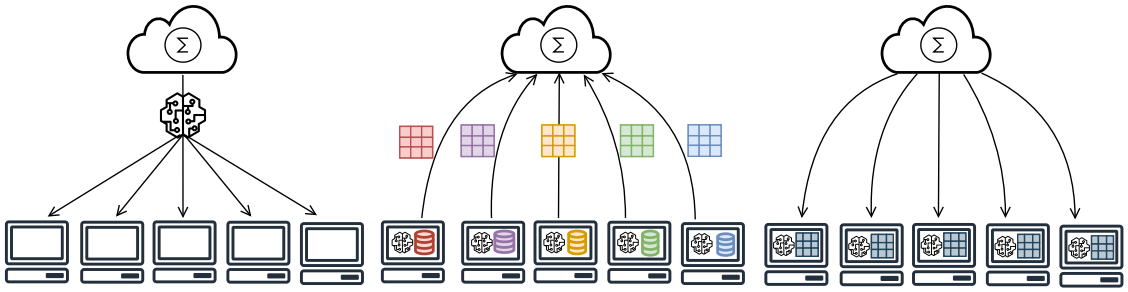

Figura 2.8. Processo de *Fine-tuning Federado* onde na Etapa 1 o servidor define o modelo pré-treinado que será utilizado no *fine-tuning*; na Etapa 2 cada cliente realiza o *fine-tuning* com seus dados locais utilizando QLoRA e compartilha apenas os *adapters* com o servidor; na Etapa 3 o servidor agrega os *adapters* recebidos e compartilha com os clientes.

hardware para ajuste de LLMs. A principal otimização é novo método de quantização dos parâmetros do modelo, consequentemente reduzindo o espaço necessário para armazenamento.

De modo geral, as otimizações realizadas pelo QLoRA incluem: *(i)* 4-*bit NormalFloat*; *(ii) Double quantization*; e *(iii) Paged Optimizers*. A quantização de 4-bits *NormalFloat* utiliza apenas 4 *bits* para representação dos parâmetros do modelo, onde 1 *bit* é utilizado para representar o sinal, 1 *bit* para expoente e 2 *bits* para mantissa. Além disso, QLoRA também aplica uma quantização de blocos, onde são atribuídas contantes distintas para cada bloco. Portanto, devido a grande quantidade de blocos para representar todo o modelo, as constantes introduzem um *overhead* adicional, assim para reduzir esse problema o QLoRA quantiza também as contantes de quantização. Assim, introduzindo apenas um *overhead* de 0.127 *bit* por parâmetro [Dettmers et al. 2023].

Por fim, o *Paged Optimizers* é uma técnica de otimização de memória que permite o treinamento em GPUs com memória limitada. Em vez de manter todos os tensores ativos na memória da GPU durante o treinamento, os *paged optimizers* gerenciam dinamicamente a movimentação de dados entre a memória da GPU (rápida, mas limitada) e a RAM do *host* (mais lenta, mas abundante), utilizando uma abordagem de *paging*, semelhante à memória virtual em sistemas operacionais. Isso possibilita o uso de otimizadores como Adam ou Adagrad mesmo em cenários de baixa VRAM, permitindo o *fine-tuning* de modelos de bilhões de parâmetros em dispositivos mais acessíveis, sem comprometer significativamente o desempenho [Dettmers et al. 2023].

2.4.4. Putting All Together

Vimos como estratégias de PEFT podem ser utilizadas para permitir o ajuste eficiente de LLMs e também como o QLora reduz ainda mais a barreira de *hardware* para o *fine-tuning* desses modelos. Nesse contexto, essas tecnologias são o alicerce para potencializar o *Fine-tuning Federado* de LLMs. Portanto, a Figura 2.8 apresenta o seguinte cenário onde clientes estão dispostos a realizar o *fine-tuning* federado de uma LLMs.

Os clientes recebem a LLM que será realizada o *fine-tuning* do servidor e realizam o *fine-tuning* utilizando utilizando QLoRA. Aqui é importante destacar que cada cliente pode utilizar configurações distintas de quantização, mas para a abordagem tradicional

dever considerar o mesmo *rank r* para as matrizes *A* e *B* do LoRA para permitir que o servidor agregue as matrizes resultantes.

Dessa forma, após realizar o *fine-tuning* da LLM cada cliente compartilha o *adapter* resultante. O compartilhamento apenas do *adapter* é essencial para reduzir o *overhead* de comunicação, uma vez que muitos menos dados são compartilhados de acordo com o *rank* definido. Em seguida, o servidor recebe o *adapters* compartilhados pelos cliente e gera um novo *adapter* contendo o conhecimento de todos os clientes, o qual é compartilhado novamente com os clientes e o processo continua até a convergência do modelo ou até atingir uma quantidade de rodadas de comunicação alvo.

Note que melhorias podem ser aplicadas nesse processo desde algoritmos de seleção de clientes para melhorar a convergência, algoritmos de clusterização para agrupar clientes com *adapters* semelhantes e gerar grupos de *adapters* personalizados. Além disso, otimizações no processo de agregação podem ser introduzidas para permitir que clientes realizem o *fine-tuning* com configurações distintas de *rank* no QLoRA de acordo com suas capacidades computacionais. O *Fine-tuning Federado* abre caminho para uma nova gama de aplicações de LLMs treinados com dados privados, melhorando a generalização do modelos e permitindo novas soluções para diversas áreas do conhecimento. A seguir serão apresentadas aplicações que podem se beneficiar do *Fine-tuning Federado* de LLMs.

2.5. Aplicações

O *Fine-tuning Federado* de modelos de linguagem tem um grande potencial para transformar diversas áreas ao unir personalização eficiente, preservação da privacidade e otimização de recursos computacionais. Dessa forma, esta seção apresenta algumas aplicações do *Fine-tuning Federado* de modelos de linguagem destacando seus objetivos e benefícios em áreas como finanças, saúde, geração e documentação de código, personalização de serviços, comunicação eficiente em sistemas distribuídos, gerenciamento adaptativo de redes, inteligência na borda da rede entre outras.

De maneira geral, os benefícios desse tipo de técnica em todas essas áreas incluem *(i)* maior privacidade, uma vez que os dados permanecem locais; *(ii)* eficiência de comunicação, reduzindo o tráfego de dados pela compactação de atualizações; e *(iii)* maior personalização, permitindo que os modelos atendam às necessidades específicas de diferentes usuários e contextos. Essas vantagens fazem do *Fine-tuning Federado* uma tecnologia essencial para aplicações que exigem equilíbrio entre personalização e segurança.

Esta seção apresenta algumas aplicações de LLMs e fine-tuning federado em diversas áreas do conhecimento, em especial as áreas relacionadas com comunicação de dados. São apresentadas aplicações da literatura e potenciais novas aplicações que demandarão soluções inovadoras.

2.5.1. Setor Financeiro

Fine-tuning de aprendizado federado possui um significativo impacto no setor financeiro, especialmente no melhoramento de análise de risco e detecção de fraude. Ao adaptar modelos para características específicas da regionalização dos mercados financeiros ou de

perfils de usuários específicos de uma determinada região, instituições financeiras podem prover serviços altamente personalizados. Por exemplo, modelos de pontuação de crédito podem ser ajustados para considerar parâmetros e características da economia local, sem precisar generalizar para todos tipos de usuários de diferentes regiões, que certamente possuem diferentes perfis. As soluções de detecção de fraudes também podem ser ajustadas para padrões específicos de atividades suspeitas, que podem variar em diferentes regiões ou países.

A principal vantagem de fine-tuning no setor financeiro é o aumento da habilidade em personalizar o modelo sem comprometer os dados sensíveis do usuários envolvidos. Modelos tradicionais de aprendizado de máquina geralmente requerem grande quantidade de dados referentes a transações financeiras para um aprendizado efetivo, possuindo riscos relacionados a privacidade dos dados. Em contrapartida, fine-tuning federado mantém os dados sensíveis na origem, garantido o cumprimento de questões de privacidade dos dados, e ainda garantindo a habilidade do modelo central de ”aprender”a partir de diversas fontes. Esse contexto provê um sistema preciso, personalizado, que preserva a privacidade dos dados ao mesmo tempo que pode ser utilizado no processo de tomada de decisão no setor financeiro.

Em [Zeng et al. 2024] os autores investigam o ajuste fino de LLM para aplicações do setor financeiro. Por meio da análise comparativa de conjuntos de dados de referência, o modelo ajustado apresentou um aumento de 2% na precisão média de resposta em vários domínios financeiros. O trabalho apresenta o ajuste do modelo tanto na infraestrutura de computação central quanto em dispositivos de borda. Em um contexto de privacidade, o trabalho [Wang et al. 2024] verifica que o ajuste fino de LLMs são cruciais para as organizações. Já no artigo “Federated Learning-Based Tokenizer for Domain-Specific Language Models in Finance” [Damoun et al. 2025] é apresentado um Tokenizador Federado de Codificação de Pares de Bytes (BPE) chamado de FedByteBPE, que utiliza uma abordagem de Aprendizado Federado (FL) para preservar a privacidade no treinamento de tokenizadores de modelos de linguagem em conjuntos de dados distribuídos. O modelo proposto vou avaliado em uma base de dados com informações financeiras.

2.5.2. Área da Saúde

Na área da saúde, o *Fine-tuning Federado* pode ser aplicado para a personalização de decisões médicas em hospitais ou consultas. A personalização das decisões e diagnósticos podem considerar características da população local ou regional, protocolo de atendimento de hospitais ou doenças raras existentes no contexto da aplicação. Por exemplo, no Brasil, um paciente com um determinado sintoma mas em diferentes regiões (Norte e Sul) pode receber um diagnóstico e/ou tratamento diferenciados. Algumas doenças são endêmicas na região Norte do Brasil, mas que raramente aparecem em outras regiões. Nesse contexto, um ajuste no modelo para considerar fatores locais ou regionais pode melhorar a qualidade do diagnóstico e tratamento de diferentes tipos de doenças.

Além disso, o *Fine-tuning Federado* garante o cumprimento de questões éticas e regulatórias ao manter dados privados locais, incluindo exames de pacientes, histórico médico ou testes de sangue realizados. Essa abordagem também incentiva a colaboração de diferentes hospitais ou centros médicos sem comprometer a confidencialidade dos da-

dos dos pacientes. Como resultado, o aprendizado federado em aplicações médicas pode ser altamente personalizado e ajustado para prover ferramentas de IA para ajudar na domada de decisão no cuidado do paciente.

O artigo "*Current research and prospects of federated language large models in the medical field*" [Tang and Deng 2024] faz um levantamento da pesquisa realizada nos últimos anos com destaque para soluções de proteção de privacidade, eficiência de dados e melhoria do desempenho do modelo. O trabalho apresenta inovações relacionadas ao pré-treinamento federado e ajuste fino dos modelos visando equilibrar o desempenho do modelo com a privacidade dos dados. Em [Sarwar 2024] os autores apresentam um fine tuning federado para ajustar LLMs para análise de saúde mental. Questões de privacidade e desempenho para o treinamento federado de LLMs no contexto biomédico é explorado em [Peng et al. 2024]. Em [Puppala et al. 2024] é proposto um chatbot personalizado para aplicações de saúde.

2.5.3. Geração e Documentação de Código

A geração e documentação de código também podem se beneficiar do fine-tuning federado. Cada pessoa, equipe de desenvolvimento ou empresas de desenvolvimento de software possuem diferentes e específicos estilo na escrita de código, padrões ou práticas de desenvolvimento. Geralmente as empresas seguem convenções internas ou linguagens de programação diferentes. Assim, um ajuste federado do modelo pode capturar e permitir uma sugestão de código mais eficiente e relacionada ao contexto da escrita do código.

É importante ressaltar que essa customização na sugestão pode ser atingida sem a exposição da base de código da empresa. Ao manter os repositórios de códigos locais, onde apenas os parâmetros do modelo são compartilhados, as organizações mantém os códigos gerados e possíveis direitos sobre eles localmente, ao mesmo tempo que a produtividade dos desenvolvedores de código aumenta. Essa abordagem vai em direção na criação de ferramentas cada vez mais personalizadas para diferentes ambientes de desenvolvimento e cultura na escrita de códigos.

Em [Weyssow et al. 2025] os autores realizam um ajuste fino com eficiência de parâmetros (PEFT) no contexto de geração de código automatizada. Técnicas de geração automatizada de códigos baseadas em modelos de linguagem enfrentam o risco de introduzir vulnerabilidades de segurança nos códigos. O trabalho "*Fine Tuning Large Language Model for Secure Code Generation*" [Li et al. 2024] apresenta um ajuste fino em LLMs para a geração de códigos. Os autores relatam que o ajuste fino aumentou em 10% a geração de código não vulnerável. Os sistemas de software têm evoluído rapidamente e, inevitavelmente, introduzido bugs em um ritmo crescente, levando a perdas significativas nos recursos consumidos pela manutenção de software. O trabalho "*When Fine-Tuning LLMs Meets Data Privacy: An Empirical Study of Federated Learning in LLM-Based Program Repair*" [Luo et al. 2024] apresenta um modelo de linguagem federado para o reparo automatizado de programas.

Nesse contexto, prevemos o ajuste fino com eficiência de parâmetros (PEFT) como uma abordagem promissora para especializar LLMs de forma eficiente para dados específicos de tarefas, mantendo um consumo razoável de recursos. Neste artigo, apresentamos um estudo abrangente de técnicas PEFT para LLMs no contexto de geração

automatizada de código.

2.5.4. Personalização de Serviços

De maneira geral, a maioria das aplicações podem se beneficiar da personalização de serviços que o fine-tuning federado é capaz de apresentar. Além das aplicações anteriores, os modelos de linguagem podem melhorar de maneira significativa assistentes virtuais, *chatbots* e sistemas de recomendação. Aspectos locais e regionais da linguagem e vocabulário, preferências culturais, e comportamentos individuais e sociais podem ser incorporados em modelos de linguagem para a criação de uma relação interação mais fiel e engajada por parte dos usuários.

A utilização do fine-tuning federado pode criar uma personalização eficiente ao mesmo tempo preservando a privacidade dos usuários e dados envolvidos no treinamento, sem a necessidade de realizar o treinamento em uma infraestrutura robusta de *hardware*. Essa é uma questão importante para empresas e organizações que operam em diferentes regiões, países ou segmentos de consumidores, onde uma solução centralizada pode não apresentar bons resultados em relação a diversidade local dos usuários. O aprendizado federado permite que o modelo evolua continuamente de maneira a melhor capturar a diversidade e necessidades dos usuários.

O trabalho [Guo et al. 2023] apresenta um modelo de visão e linguagem pré-treinados para a captura de características latentes de usuários. Utilizando uma abordagem federada e ajustes, o trabalho apresenta o pFedPrompt, um modelo totalmente personalizado e alinhado às características locais do usuário. Em [Yang et al. 2023] os autores apresentam um modelo federado e personalizados para a geração de prompts específicos para cada cliente. O trabalho aborda a heterogeneidade dos dados dos clientes como motivação para a criação de um modelo de linguagem personalizado. No trabalho "Fine-Tuning Personalization in Federated Learning to Mitigate Adversarial Clients" [Allouah et al. 2024] apresenta um modelo federado de linguagem com ajustes finos que contorna o problema de colaboração dos clientes na presença de clientes maliciosos. A personalização dos modelos mitiga a presença de clientes maliciosos no processo de treinamento federado.

2.5.5. Comunicação Eficiente em Sistemas Distribuídos

Em sistemas distribuídos, especialmente Internet das Coisas, redes de sensores sem fio, e arquiteturas de computação multi-acesso na borda da rede (multi-access edge computing – MEC), a comunicação eficiente é fundamental para a execução de diversas aplicações. Nesse contexto, ajustes de modelos federados podem permitir adaptações locais de modelos responsáveis para a geração de mensagens, compressão, *parsing* e interpretação de imagens. Ao realizar fine-tuning de modelos que estão próximos aos dispositivos da rede, é possível criar protocolos de comunicação ou formatos de transferência de mensagens que se mostram melhores em cenários com certas restrições de rede, como baixa largura de banda, alta latência, comunicação intermitente ou capacidades heterogêneas dos dispositivos.

Por exemplo, um ajuste de modelo na geração de mensagens de um dispositivo em execução em um ambiente rural com baixa largura de banda pode priorizar/forma-

tar/interpretar mensagens de maneira diferente em comparação a um ambiente 5G com alta largura de banda. A comunicação adaptativa pode reduzir o volume de dados trocados entre os dispositivos da rede de maneira dinâmica dependendo do contexto e recursos de computação e comunicação. Além disso, ao aplicar uma técnica federada no ajuste dos modelos, a adaptação pode ter atingida sem a centralização dos dados coletados pelos dispositivos, mitigando problemas de segurança e privacidade de informação. Em um cenário de redes 6G, onde espera-se que bilhões de dispositivos estejam conectados, o fine-tuning federado de modelos de linguagem pode ter um papel importante na eficiência da comunicação onde diversos dispositivos com diferentes recursos computacionais e de comunicação estão interligados.

Em [Guo et al. 2025] os autores apresentam e estudo o problema de modelos de linguagens e comunicação semântica. O trabalho apresenta uma discussão sobre a capacidade de LLMs para definir a perda semântica na comunicação e o trabalho apresenta um método para quantificar a importância semântica de uma palavra/quadro na comunicação de forma eficiente e considerando recursos limitados de comunicação. A comunicação semântica eficiente em redes IoT utilizando modelos de linguagens também é estudada em [Kalita 2024]. A autora argumenta que, ao extrair significado sobre uma mensagem a ser transmitida, a mensagem pode ser decodificada mesmo na presença de erros na comunicação de dados. Outros trabalhos [Zhao et al. 2024, Qiyang Zhao 2024, Nguyen et al. 2024] também abordam problemas similares.

No contexto de redes sem fio, o trabalho [Sun et al. 2025] apresenta um framework chamado *AirFL-LoRA (Wireless Over-the-Air Federated Learning based Low-Rank Adaptation)*. O AirFL-LoRA tem como objetivo permitir o fine-tuning eficiente de modelos de linguagem sobre redes sem fio. O trabalho apresenta os princípios fundamentais para aplicar LLMs ao domínio de redes sem fio, que possui o potencial de otimizar a comunicação e eficiência em sistemas distribuídos.

2.5.6. Gerência Adaptativa de Redes

Em ambientes de redes dinâmicos, o fine-tuning federado de modelos de linguagem pode ajudar a personalizar o gerenciamento de redes considerando diferentes topologias de redes, códigos para gerenciamento, padrões no tráfego e ameaças de segurança. Por exemplo, o ajuste de modelo de linguagem pode ser realizado localmente para o reconhecimento de comportamentos normais em diferentes segmentos de uma rede, melhorando a acurácia na detecção de ameaças sem a centralização de todas as informações transmitidas na rede. Essa aplicação é particularmente importante em cenários de cidades inteligentes, onde diferentes tipos de redes com diferentes características e objetivos devem interoperar para o correto funcionamento das diferentes aplicações.

No artigo "Enhancing Network Management Using Code Generated by Large Language Models" [Mani et al. 2023] os autores apresentam uma nova abordagem que permite experiências de gerenciamento de rede baseadas em linguagem natural, aproveitando grandes modelos de linguagem (LLMs) para gerar código específico de tarefa a partir de consultas em linguagem natural. Em [Lira et al. 2024] os autores apresentam um gerador de configuração de rede que arquiteta agentes de configuração por grandes modelos de linguagem. O LLM-NetCFG pode gerar configurações automaticamente, ve-

rificá-las e configurar dispositivos de rede com base em intenções expressas em linguagem natural.

2.5.7. Inteligência na Borda

Com o crescimento de diversas aplicações em sistemas distribuídos, incluindo a comunicação e computação na borda da rede, o fine-tuning federado de modelos de linguagens pode suportar novos serviços inteligentes que estão localizados mais próximos aos usuários. Os modelos em execução nos dispositivos da borda podem ser ajustados para o contexto e ambiente local, considerando a interação com o usuário ou aplicações que não demandem uma uma comunicação com servidores centrais. Isso reduz o custo de comunicação e minimiza a latência na troca de mensagens. Além disso, a utilização de modelos de linguagem na borda da rede tem o potencial de prover o desenvolvimento de aplicações mais autônomas e resilientes com o avanço das redes 5G/6G, onde a comunicação direta e constante com servidores centrais pode não estar sempre disponível.

O trabalho [Friha et al. 2024] faz uma levantamento bibliográfico sobre LLM e inteligência na borda, abordando, entre outros tópicos, o aprendizado federado e fine-tuning de modelos de linguagem. Aprendizado federado e aprendizado na borda para modelos de linguagem são discutidos em [Piccialli et al. 2025]. O artigo tem como objetivo fornecer uma análise aprofundada do estado atual do treinamento e implantação de LLMs eficientes em ambientes federados e na borda da rede. No artigo "Federated Fine-Tuning of LLMs on the Very Edge: The Good, the Bad, the Ugly" [Woisetschläger et al. 2024]. Os autores investigam os desafios e oportunidades do fine-tuning federados de LLMs em dispositivos na borda, considerando restrições de recursos computacionais, eficiência energética e largura de banda. Os autores ainda mostram resultados de experimentos em testbed reais com dispositivos NVIDIA Jetson AGX Orin. Como conclusão, os dispositivos de borda com restrição de recursos, como baixo poder computacional, largura de banda e disponibilidade de energia apresentam desafios adicionais na aplicação de LLMs em um ambiente federado. Os autores enumeram o fine-tuning de modelos como uma estratégia para a criação de sistemas eficientes.

2.6. Hands-On - *Fine-tuning Federado* de LLMs

Para solidificar os conceitos apresentados anteriormente, esta seção apresenta o fluxo de ações realizadas pelo *framework* Flower [Beutel et al. 2020] para efetivação do aprendizado. O código apresentado neste documento restringe-se à assinatura de um método a ser implementado; a implementação completa está disponível no GitHub[2]. O objetivo da implementação é exemplificar a adoção de LLMs em um problema de redes de computadores e por que o ajuste fino é necessário.

2.6.1. Ambiente de desenvolvimento

Neste minicurso utilizaremos o *Google Colaboratory (Colab)*, que é uma plataforma gratuita baseada em nuvem para escrever e executar código *Python* no seu navegador. Ele oferece acesso gratuito a *GPUs/TPUs* e se integra ao *Google Drive*.

[2]`https://github.com/AllanMSouza/MC2-SBRC2025`

Para começar, acesse o site do *Google Colab*[3] e faça login clicando no botão azul no canto superior direito da página. Você precisa ter uma conta do *Google* para usar esta ferramenta. Após fazer login, você verá uma caixa pop-up semelhante à mostrada na Figura 2.9. Nela, será possível criar um *Notebook* onde o código será desenvolvido ao clicar em "Novo Notebook".

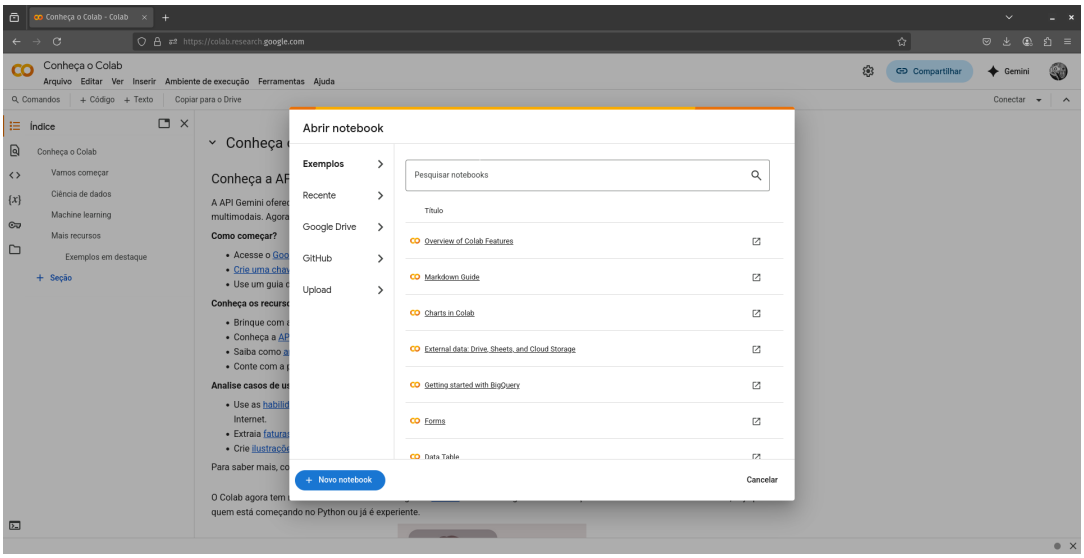

Figura 2.9. Tela inicial do *Colab*.

Após executar a ação acima, um notebook como apresentado na Figura 2.10 será disponibilizado. A organização de um *Notebook* é definida através de células. As células textuais demarcam seções e realizam uma documentação mais detalhada da programação. As células de código executam os comandos desejados pelo usuário.

2.6.2. Projeto da simulação

O Flower segue a arquitetura de referência acima apresentada. O seu projeto é orientado a objetos e define três classes principais, responsáveis pelo treinamento distribuído:

1. **Servidor**: coordena o fluxo de ações a serem realizadas por uma estratégia e gerencia a comunicação com os clientes para execução do treinamento federado;
2. **Estratégia**: é o algoritmo de aprendizado federado executado no servidor. A estratégia decide como selecionar clientes, configurá-los para treinamento, agregar atualizações e avaliar modelos;
3. **Cliente**: define como o treinamento/avaliação local será executado e permite que o servidor os execute remotamente.

O relacionamento entre elas ocorre em uma sequência de atividades bem definida. A Figura 2.11 ilustra a sequência, que também será utilizada como ordem das seções seguintes que explicam que implementações devem ser realizadas para viabilizar o ajuste fino federado de LLMs.

[3]`https://colab.research.google.com/`

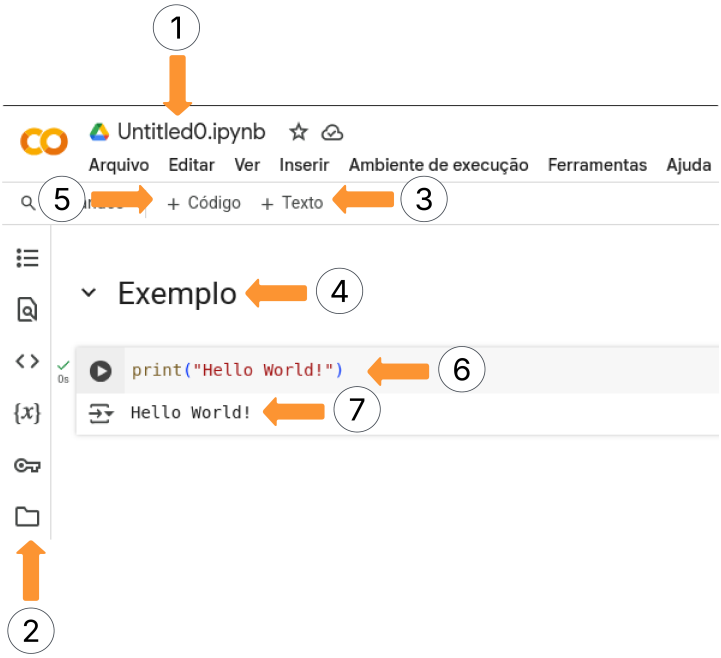

Figura 2.10. *Notebook*: 1) Nome a ser definido 2) Acesso ao Drive 3) Adição de célula de texto 4) Exemplo de célula de texto 5) Adição de célula de código 6) Exemplo de código *Python* 7) Saída da execução da célula.

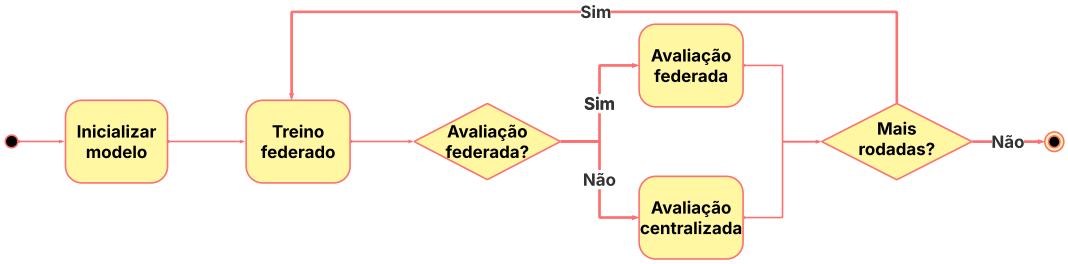

Figura 2.11. Sequência de atividades do aprendizado federado.

2.6.3. Inicialização do modelo

Esta atividade requer a interação entre o objeto servidor e o objeto estratégia, com o objetivo de disponibilizar ao servidor os parâmetros iniciais do modelo a ser treinado. A Figura 2.12 ilustra, através de um diagrama de sequência, a interação.

Brevemente após sua inicialização, o objeto servidor chama uma única vez o método *initialize_parameters* do objeto estratégia, a fim de obter os parâmetros iniciais do modelo a ser treinado. Esses parâmetros foram passados ao objeto estratégia em sua inicialização. Caso a estratégia não tenha recebido os parâmetros em sua inicialização, o método retorna *None* e o servidor irá obter os parâmetros iniciais do modelo de algum cliente. Para o caso do ajuste fino de LLMs, uma boa prática é sempre inicializar o objeto estratégia com os parâmetros iniciais do modelo. O método *initialize_parameters* segue a estrutura seguinte:

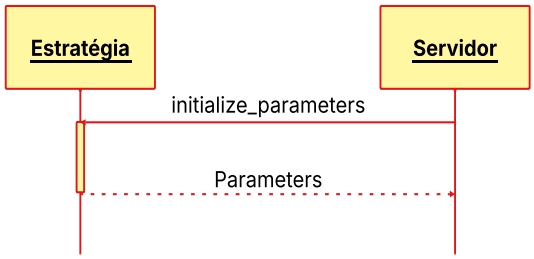

Figura 2.12. Diagrama de sequência: inicialização do modelo.

```python
def initialize_parameters(
        self, client_manager: ClientManager
    ) -> Optional[Parameters]:
        # Implementação aqui

        return initial_parameters
```

O servidor passa um objeto do tipo *ClientManager* para a estratégia, caso ela queira utilizar informações sobre os clientes conectados ao servidor na inicialização. O retorno do tipo *Parameters* garante ao servidor os pesos do modelo a serem modificados no treinamento.

2.6.4. Treino federado

A interação desta atividade envolve os objetos das três classes acima apresentadas: servidor, estratégia e cliente. A interação entre eles tem por objetivos: selecionar os clientes participantes da rodada de comunicação, o envio do modelo global para treinamento nos clientes, o treinamento local, o envio dos modelos locais ao servidor e a agregação em um novo modelo global. A sequência de interações entre os objetos, métodos a serem implementados e objetos comunicados entre as classes está representada na Figura 2.13.

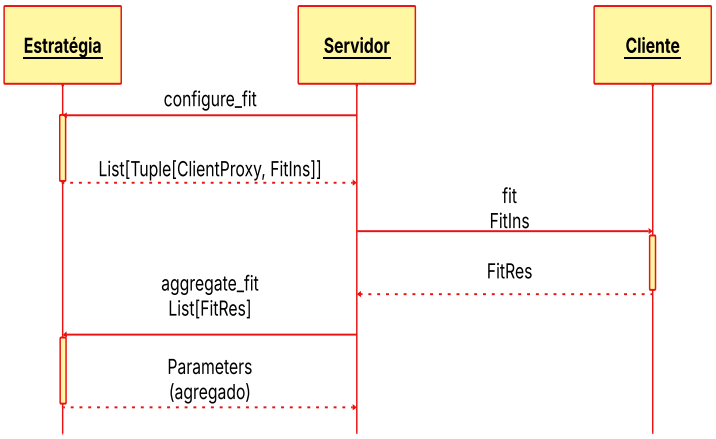

Figura 2.13. Diagrama de sequência: treinamento do modelo.

Primeiramente, há a chamada pelo servidor do método *configure_fit* da estratégia.

Ele configura a próxima rodada de treinamento, selecionando os clientes e instruindo o treinamento. A assinatura deixa clara a responsabilidade desse método.

```
def configure_fit(
    self, server_round: int, parameters: Parameters,
     ↪  client_manager: ClientManager
) -> List[Tuple[ClientProxy, FitIns]]:
    # Implementação aqui

    # Retorna par cliente/config
    return [(client, fit_ins) for client in selected_clients]
```

Note que o retorno é uma lista de tuplas contendo as instruções que serão enviadas para cada cliente selecionado. A implementação desse método geralmente envolve:

- Utilizar o objeto client_manager para realizar a seleção de clientes, seja de maneira aleatória ou seguindo algum critério. Este segundo caso requer a implementação de uma subclasse de *ClientManager*.
- Definir para cada cliente selecionado, que são instâncias da classe *ClientProxy*, um objeto FitIns que conterá as instruções que norteiam o treinamento no cliente.

Cada cliente selecionado executa o método *fit*, apresentado no código abaixo.

```
def fit(self, parameters, config):
        # Implementação aqui
        return get_parameters(self.net),
         ↪  len(self.trainloader), {}
```

O objetivo dessa ação no cliente é atualizar o modelo local com os parâmetros do modelo global recebidos do servidor, representado por *parameters*, treinar o modelo seguindo as orientações contidas em *config* e retornar para o servidor um objeto *FitRes*, contendo uma tupla de valores. O primeiro valor dessa tupla são os parâmetros do modelo local atualizado após treino, o segundo é o tamanho do *dataset* local utilizado no treinamento e o terceiro é um dicionário de métricas úteis à seleção, agregação e demais ações da estratégia que otimizam o aprendizado.

O servidor, após receber dos clientes os objetos *FitRes*, chama da estratégia o método de agregação *aggregate_fit*, passando como argumento as informações recebidas dos clientes.

```python
def aggregate_fit(
    self,
    server_round: int,
    results: List[Tuple[ClientProxy, FitRes]],
    failures: List[Union[Tuple[ClientProxy, FitRes],
     ↪  BaseException]],
) -> Tuple[Optional[Parameters], Dict[str, Scalar]]:
    #Implementação aqui
    return parameters_aggregated, metrics_aggregated
```

Como falhas no treinamento do cliente e de comunicação da resposta para o servidor podem ocorrer, *aggregate_fit* recebe tanto a lista de resultados como a de falhas. Então, o método realiza a agregação dos parâmetros dos modelos locais e de métricas, que formam sua tupla de retorno. Caso não haja resultados em número suficiente para atualizar o modelo global, as variáveis retornadas devem conter valor *None*.

2.6.5. Avaliação

A atividade em sequência do treinamento é a avaliação do modelo global agregado. Ela pode ocorrer de duas formas não mutuamente excludentes: centralizada e federada. A Figura 2.14 apresenta o diagrama de sequência de ações realizadas por ambas avaliações.

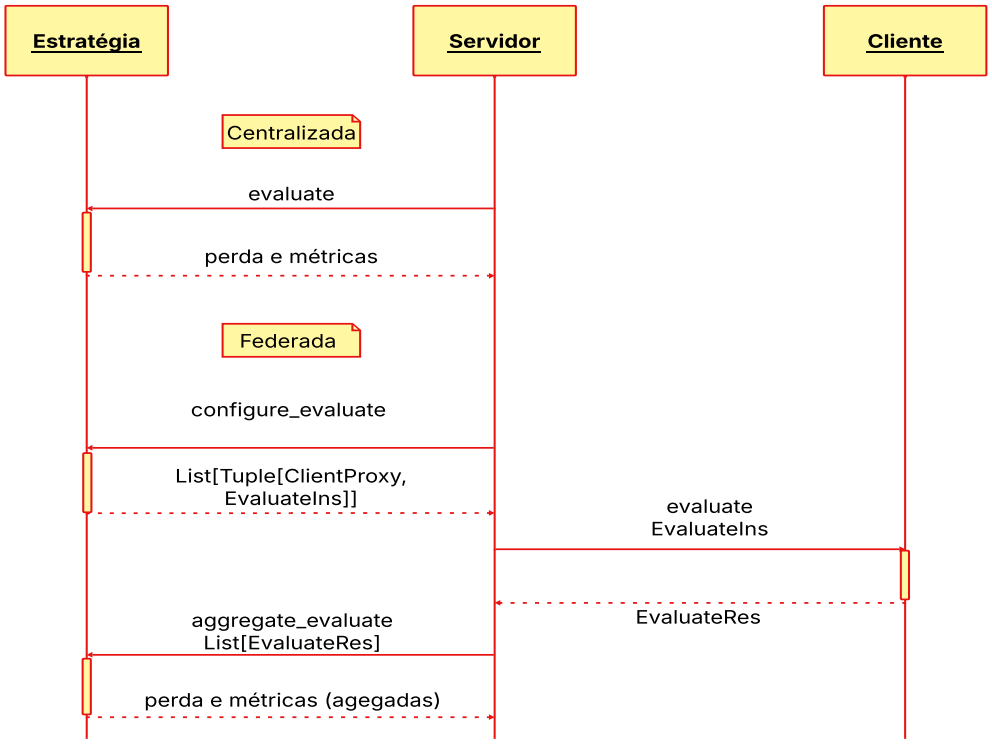

Figura 2.14. Diagrama de sequência: avaliação do modelo.

A Avaliação Centralizada, também conhecida como avaliação do lado do servidor, é um conceito bastante simples. Ela funciona de maneira semelhante à avaliação em aprendizado de máquina centralizado. Se tivermos um conjunto de dados armazenado no servidor que possa ser utilizado para avaliação, isso é ótimo. Podemos analisar o modelo que foi atualizado após cada sessão de treinamento sem precisar enviá-lo para os clientes. Além disso, temos a vantagem de ter acesso ao nosso conjunto completo de dados de avaliação sempre que precisamos. Nela o objeto servidor chama do objeto estratégia o método *evaluate*. Ele recebe os parâmetros do modelo global e realiza o teste nesse modelo utilizando um *dataset* auxiliar, localizado no servidor. O retorno do método é uma tupla contendo: perda do modelo no *dataset* de teste e um dicionário de outras métricas definido pelo usuário.

```
def evaluate(self, parameters: Parameters) ->
    Optional[Tuple[float, Dict[str, Scalar]]]:
    # Implementação aqui
    return loss, metrics
```

O retorno é opcional. A estratégia pode não implementar a avaliação centralizada ou porque o método de avaliação pode não ser concluído com sucesso. Por exemplo, devido a uma falha.

A Avaliação Federada, ou avaliação do lado do cliente, é mais complexa, mas também oferece mais benefícios. Ela não depende de um conjunto de dados centralizado e nos permite avaliar modelos em uma quantidade maior de dados, o que geralmente resulta em avaliações mais precisas. De fato, em muitos casos, precisamos usar a Avaliação Federada para conseguir resultados representativos. No entanto, essa abordagem tem suas desvantagens. Quando começamos a avaliar do lado do cliente, é importante entender que nosso conjunto de dados pode mudar ao longo das diferentes fases de aprendizado, especialmente se os clientes não estiverem sempre disponíveis. Além disso, os dados que cada cliente possui também podem variar com o tempo. Isso pode resultar em avaliações instáveis; assim, mesmo sem modificarmos o modelo, podemos observar flutuações nos resultados ao longo das diversas fases de avaliação.

A sequência de ações da avaliação federada é similar ao do treinamento federado. O objeto servidor chama o método de configuração da avaliação, *configure_evaluate*, que seleciona os clientes e configura como deve ser realizada a avaliação por cada cliente selecionado.

```
def configure_evaluate(
    self, server_round: int, parameters: Parameters,
     client_manager: ClientManager
) -> List[Tuple[ClientProxy, EvaluateIns]]:
    # Implementação aqui

    # Retorna par cliente/config
    return [(client, evaluate_ins) for client in
     selected_clients]
```

Assim como a seleção de clientes para treinamento, a seleção de clientes para

avaliação pode usufruir de uma lógica mais sofisticada de seleção.

Cada cliente selecionado, ao receber do servidor os parâmetros do modelo global e a configuração a ser utilizada na avaliação local, atualiza um modelo local com os parâmetros, configura o teste e realiza-o. Essa sequência de ações é implementada no método *evaluate* do objeto cliente.

```python
def evaluate(
        self, parameters: NDArrays, config: dict[str, Scalar]
    ) -> tuple[float, int, dict[str, Scalar]]:
    # Implementação aqui
    return float(loss), len(self.valloader), {}
```

O retorno do cliente é um objeto *EvaluateRes*, que contém uma tupla de informações, similar ao retorno do treinamento. Porém, os valores contidos na tupla são: a perda na avaliação local, a quantidade de dados utilizada na avaliação e um dicionário de métricas definidas pelo usuário para auxiliar o processo.

Também similar à agregação do treinamento, a agregação da avaliação federada, *agregate_evaluate*, recebe dos clientes resultados ou falhas. Caso o número de falhas seja elevado, impedindo uma análise do desempenho realista, o método retorna *None*. Caso contrário, a agregação dos resultados é realizada e retornam-se para o servidor as perdas e demais métricas definidas pelos usuários de forma agregada.

```python
def aggregate_evaluate(
    self,
    server_round: int,
    results: List[Tuple[ClientProxy, EvaluateRes]],
    failures: List[Union[Tuple[ClientProxy, FitRes],
     ↪  BaseException]],
) -> Tuple[Optional[float], Dict[str, Scalar]]:
    # Implementação aqui

    return loss_aggregated, metrics_aggregated
```

2.7. Desafios e Oportunidades

Esta seção apresenta os principais desafios e oportunidades das tecnologias abordadas no minicurso, o objetivo principal é direcionar os participantes à questões em aberto que ainda precisam ser endereçadas, criando novas linhas e grupos de pesquisa nas áreas de redes de comunicação, cidades inteligentes e Internet das Coisas (IoT).

2.7.1. Desafios

O *Fine-tuning Federado* de modelos de linguagem apresenta desafios significativos que ainda demandam atenção em pesquisas.

2.7.1.1. Heterogeneidade de Dados

A heterogeneidade de dados é um dos principais desafios no contexto do *Fine-tuning Federado*, pois os dados utilizados em cada nó (dispositivo cliente) geralmente possuem distribuições diferentes [Bai et al. 2024]. Isso vai de encontro com os métodos tradicionais de treinamento centralizado, onde os dados podem ser organizados e balanceados. No cenário federado, tais diferenças podem levar a um modelo global que tem dificuldade em generalizar bem para todos os contextos.

Esse problema é conhecido como dados *non-IID* (dados não independentemente e identicamente distribuídos) e pode prejudicar tanto a estabilidade quanto a convergência do treinamento. Clientes com dados altamente específicos podem direcionar o modelo para direções inconsistentes com o objetivo global [Zhu et al. 2021]. Estratégias como reponderação de atualizações, agrupamento de clientes similares e *clustering* de modelos parciais são propostas na literatura como alternativas para mitigar esses efeitos adversos [Yan et al. 2025].

2.7.1.2. Heterogeneidade Sistêmica

A heterogeneidade sistêmica refere-se às diferenças entre os dispositivos clientes participantes do treinamento federado, como capacidade de memória, poder computacional, largura de banda e conectividade [Singh and Adhikari 2025]. No *Fine-tuning Federado*, que exige recursos consideráveis, essa variação se torna crítica. Enquanto alguns dispositivos podem processar grandes *batches* e armazenar múltiplas camadas do modelo, outros podem ser severamente limitados.

Essa diferença entre os dispositivos pode causar atrasos ou bloqueios na agregação global, pois o sistema precisa aguardar atualizações de dispositivos mais lentos ou menos responsivos. Além disso, pode comprometer a eficiência geral do treinamento ao forçar a adoção de configurações que priorizem a compatibilidade com os dispositivos mais fracos [Li et al. 2025] ou, excluir os dispositivos com menor capacidade, dificultando uma inclusão equitativa.

2.7.1.3. *Memory Wall*

No contexto do *Fine-tuning Federado* de LLMs, o fenômeno conhecido como *memory wall* refere-se às limitações de memória presentes nos dispositivos participantes, que frequentemente não conseguem armazenar ou processar eficientemente modelos de grande porte [Wu et al. 2025b]. Dado o tamanho expressivo das LLMs modernas, que podem facilmente ultrapassar centenas de megabytes ou mesmo *gigabytes*, muitos dispositivos locais, como *smartphones* ou sensores embarcados, simplesmente não possuem memória suficiente para carregar o modelo completo.

Essa limitação de memória impede que esses dispositivos contribuam de forma plena com seus dados locais para o treinamento federado [Tam et al. 2024]. Como resultado, dados potencialmente valiosos, e muitas vezes representativos de domínios específicos ou contextos particulares, deixam de ser utilizados na atualização do modelo

global. Isso compromete tanto a diversidade do aprendizado quanto a representatividade do modelo final.

2.7.1.4. Sobrecarga de Comunicação

A sobrecarga de comunicação é um dos gargalos mais significativos no treinamento federado de LLMs [Tao et al. 2025]. A cada rodada de treinamento, os clientes precisam enviar atualizações de gradientes ou dos próprios parâmetros do modelo para o servidor central, o que pode implicar em transferências de dezenas ou centenas de *megabytes* por cliente. Em ambientes com conexão instável ou largura de banda limitada, isso compromete fortemente a eficiência do sistema.

Além disso, o custo acumulado dessas transferências se torna ainda mais expressivo à medida que o número de clientes aumenta, impactando diretamente a escalabilidade do sistema [Wu et al. 2024a]. Em cenários com centenas ou milhares de dispositivos participantes, a largura de banda necessária para sustentar múltiplas rodadas de comunicação simultânea pode exceder a capacidade da infraestrutura de rede disponível. Isso não apenas introduz latências significativas, como também aumenta o risco de falhas de sincronização entre clientes e servidor, comprometendo a estabilidade e a qualidade do treinamento global.

2.7.1.5. Sobrecarga de Computação

O *fine-tuning* de LLMs envolve grande intensidade computacional, especialmente quando se lida com modelos com bilhões de parâmetros. Em cenários federados, esse custo precisa ser absorvido por dispositivos heterogêneos e muitas vezes com capacidade limitada. Isso pode inviabilizar a participação de diversos clientes, além de introduzir atrasos significativos nas rodadas de treinamento.

A sobrecarga computacional decorre não apenas do número de parâmetros, mas também da complexidade dos cálculos envolvidos, como operações de atenção e *backpropagation* em estruturas profundas [Zhang and You 2024]. Para contornar esse obstáculo, surgem soluções como o uso de *Parameter Efficient Fine-Tuning (PEFT)*, com destaque para métodos como LoRA (*Low-Rank Adaptation*), *adapters* e *prompt tuning*, que reduzem drasticamente a quantidade de parâmetros treináveis por cliente [Kuang et al. 2024].

2.7.2. Oportunidades

2.7.2.1. Aplicações em Domínios Sensíveis

Uma das maiores oportunidades do *fine-tuning* federado reside na sua aplicação em domínios sensíveis, como saúde, finanças e segurança, onde a privacidade dos dados é essencial. A capacidade de adaptar modelos poderosos a contextos específicos mantendo a privacidade abre espaço para a adoção ampla em setores altamente regulados.

No setor financeiro, por exemplo, a aplicação do *Fine-tuning Federado* permite a customização de modelos de análise de risco e detecção de fraudes considerando especi-

ficidades regionais e padrões locais de comportamento. Isso viabiliza uma maior precisão sem expor informações sensíveis dos usuários, o que é essencial frente às exigências legais como a LGPD. Contudo, a principal dificuldade nesse contexto é garantir a eficácia do modelo em cenários de alta heterogeneidade de dados, considerando que os perfis financeiros variam amplamente entre regiões, tornando os dados não independentes e não identicamente distribuídos (non-IID) [Damoun et al. 2025]. Além disso, as instituições operam sob diferentes legislações e regulamentos, o que impõe limitações técnicas à interoperabilidade dos modelos treinados. Outro aspecto desafiador é o risco de vazamento indireto de informações durante o processo de agregação de atualizações, o que exige o uso de técnicas robustas de segurança e anonimização. Contudo, apesar dos desafios, este cenário apresenta uma grande oportunidade para personalizar modelos que atendam com precisão demandas regionais específicas, melhorando significativamente a eficiência operacional e a confiança em instituições financeiras.

2.7.2.2. Personalização de Assistentes Virtuais

A personalização de assistentes virtuais e modelos de linguagem por meio de *Fine-tuning Federado* caracteriza-se por uma abordagem avançada de adaptação de modelos pré-treinados, na qual a atualização de parâmetros ocorre localmente, diretamente nos dispositivos dos usuários. Nesse paradigma, cada nó computacional (tipicamente um *smartphone*) realiza treinamento local utilizando seus próprios dados, que refletem características linguísticas particulares, como variações de dialeto, jargões regionais e padrões de interação específicos. Como qualquer outro método federado, ele emprega algoritmos de agregação, em que os gradientes ou as atualizações de parâmetros provenientes de cada nó são combinados para aprimorar um modelo global, permitindo que a heterogeneidade dos dados, inerente ao cenário não-iid (não independente e identicamente distribuído), seja adequadamente considerada.

Por se tratarem de dados pessoais dos usuários, pode-se utilizar técnicas de privacidade diferencial e mecanismos de criptografia durante o processo de comunicação entre os dispositivos e o servidor central para oferecer garantias robustas de preservação da privacidade dos dados. Essa estratégia possibilita que informações sensíveis permaneçam localmente, atendendo a requisitos regulatórios e assegurando a confidencialidade dos dados individuais, sem impactar negativamente o desempenho do modelo global. Do ponto de vista computacional, o *Fine-tuning Federado* se mostra altamente relevante para dispositivos com recursos limitados, permitindo o uso de técnicas como o *Parameter Efficient Fine-Tuning (PEFT)* para reduzir o *overhead* computacional e o energético, aspectos críticos em ambientes móveis.

2.7.2.3. Aprendizado Contínuo e Coleta Contínua de Dados

A integração de aprendizado contínuo ao *Fine-tuning Federado* possibilita a atualização progressiva dos modelos à medida que novos dados são coletados localmente, promovendo uma adaptação constante a mudanças nos padrões de uso, comportamento dos usuários ou contexto operacional. Esse paradigma é especialmente relevante em

aplicações que operam em ambientes dinâmicos, como dispositivos móveis, sensores embarcados e plataformas digitais em evolução. A coleta contínua de dados permite capturar variações temporais e eventos inesperados, mantendo a relevância dos modelos ao longo do tempo, sem exigir recomeços frequentes de treinamento. Contudo, essa abordagem impõe desafios como o fenômeno da catástrofe do esquecimento, em que o modelo tende a perder desempenho em tarefas anteriores. Estratégias como buffers de memória, regularização baseada na importância de parâmetros e modularização de redes são empregadas para mitigar esse efeito.

Do ponto de vista da privacidade, a coleta contínua de dados impõe a necessidade de mecanismos robustos de proteção, visto que o volume e a sensibilidade das informações aumentam com o tempo. Técnicas como privacidade diferencial online, agregação segura e aprendizado assíncrono são fundamentais para preservar a confidencialidade das atualizações em fluxo. Além disso, o uso de estratégias de treinamento federado, que determinam quando iniciar um novo ciclo com base na qualidade ou quantidade dos dados recém-coletados, permite otimizar recursos computacionais e de comunicação, algo crítico em dispositivos com restrições de energia e conectividade. Dessa forma, o *Fine-tuning Federado* com aprendizado e coleta contínuos viabiliza o desenvolvimento de sistemas que evoluem de forma autônoma, mantendo-se atualizados, personalizados e seguros em ambientes reais em constante transformação.

2.8. Conclusão

O avanço dos LLMs representa um dos marcos mais impactantes da inteligência artificial. Esses modelos, capazes de compreender e gerar linguagem com elevado grau de sofisticação, tornaram-se rapidamente centrais em aplicações diversas da medicina à indústria de software, passando por educação, comunicação, segurança e finanças. No entanto, o real potencial desses modelos só é liberado quando eles são ajustados para contextos específicos, por meio do processo conhecido como *fine-tuning*.

Ao longo deste minicurso, exploramos em profundidade como o *fine-tuning* federado emerge como uma resposta eficiente aos principais obstáculos da especialização de LLMs: privacidade de dados, custo computacional elevado, distribuição heterogênea de dados e dispositivos, e demandas por personalização e escalabilidade. O FL, ao permitir que o ajuste dos modelos ocorra localmente (mantendo os dados sensíveis nos dispositivos de origem) proporciona uma arquitetura distribuída que respeita a privacidade e otimiza o uso dos recursos existentes.

Mais do que uma solução prática, o *fine-tuning* federado representa uma mudança de paradigma no desenvolvimento e na aplicação de modelos de linguagem. Ele desloca o eixo centralizador da IA tradicional para uma abordagem cooperativa, onde diferentes agentes contribuem para um modelo global sem jamais abrir mão do controle sobre seus dados. Essa descentralização é não apenas uma alternativa técnica, mas um imperativo ético e regulatório em diversos setores.

Para tornar essa visão viável, técnicas como LoRA e QLoRA são fundamentais. Elas permitem que o ajuste fino seja realizado com alterações mínimas nos parâmetros do modelo, reduzindo drasticamente os custos de memória, processamento e comunicação. A combinação dessas abordagens com arquiteturas federadas viabiliza o *fine-tuning*

mesmo em dispositivos de borda, como *smartphones* ou sensores embarcados, democratizando o acesso à IA avançada.

Durante o minicurso, mostramos como o *fine-tuning* federado pode ser aplicado em diversos cenários de alto impacto: Na saúde, para diagnóstico personalizado respeitando dados sensíveis; Nas finanças, para detecção de fraudes e análise de crédito em nível regional; No desenvolvimento de software, para geração e documentação de código adaptado aos padrões de cada equipe; Na comunicação eficiente, para compressão e interpretação semântica em redes distribuídas e restritas; Na borda da rede, para serviços inteligentes com baixa latência e autonomia local.

Além disso, discutimos os principais desafios técnicos em aberto, como a heterogeneidade de dados (non-IID), a heterogeneidade sistêmica entre os dispositivos participantes, a limitação de memória (*memory wall*), e a sobrecarga de comunicação e computação. Também refletimos sobre as oportunidades futuras, especialmente em setores sensíveis e na personalização de serviços, como assistentes virtuais, sistemas de recomendação e interfaces naturais de linguagem.

O conteúdo prático apresentado no minicurso, com o uso do *framework* Flower, permitiu aos participantes experimentarem na prática os conceitos de treinamento federado com LLMs, consolidando os aspectos teóricos por meio de simulações concretas. Essa integração entre teoria e prática é essencial para fomentar a criação de soluções reais, sustentáveis e escaláveis.

Com base nas discussões desenvolvidas, algumas direções futuras se destacam como especialmente promissoras: Algoritmos avançados de agregação: que permitam maior flexibilidade na combinação de modelos ajustados localmente, mesmo em condições de dados heterogêneos e infraestrutura assimétrica. *Federated Evaluation*: estratégias robustas para avaliar modelos federados de forma eficiente e representativa, respeitando os limites de privacidade e variabilidade dos dados locais. Automação do *fine-tuning* federado: com sistemas que escolham automaticamente os melhores métodos de PEFT e estratégias de comunicação adaptadas à capacidade de cada cliente. Interoperabilidade e padronização: para permitir a integração entre modelos treinados em diferentes organizações e setores, respeitando normas técnicas e legais.

Em resumo, o *fine-tuning* federado representa não apenas uma evolução tecnológica, mas um novo modelo para pensar inteligência artificial em larga escala uma IA que respeita a diversidade, protege a privacidade, e distribui valor de forma equitativa. Trata-se de um campo emergente, com vasto potencial para inovação, impacto social e transformação de processos em todos os níveis da sociedade. Este minicurso buscou capacitar seus participantes para não apenas compreender esse cenário, mas também para atuar ativamente em sua construção. O futuro da IA é colaborativo, e o *fine-tuning* federado é um de seus pilares centrais.

Referências

[Aghajanyan et al. 2020] Aghajanyan, A., Zettlemoyer, L., and Gupta, S. (2020). Intrinsic dimensionality explains the effectiveness of language model fine-tuning.

[Allouah et al. 2024] Allouah, Y., El Mrini, A., Guerraoui, R., Gupta, N., and Pinot, R.

(2024). Fine-tuning personalization in federated learning to mitigate adversarial clients. In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C., editors, *Advances in Neural Information Processing Systems*, volume 37, pages 100816–100844. Curran Associates, Inc.

[Bai et al. 2024] Bai, J., Chen, D., Qian, B., Yao, L., and Li, Y. (2024). Federated fine-tuning of large language models under heterogeneous language tasks and client resources. *arXiv e-prints*, pages arXiv–2402.

[Beutel et al. 2020] Beutel, D. J., Topal, T., Mathur, A., Qiu, X., Fernandez-Marques, J., Gao, Y., Sani, L., Kwing, H. L., Parcollet, T., Gusmão, P. P. d., and Lane, N. D. (2020). Flower: A friendly federated learning research framework. *arXiv preprint arXiv:2007.14390*.

[Brave 2015] Brave (2015). O navegador que coloca você em primeiro lugar — Brave — brave.com. `https://brave.com/pt-br/`. [Accessed 07-04-2025].

[Capanema et al. 2025] Capanema, C. G. S., de Souza, A. M., da Costa, J. B. D., Silva, F. A., Villas, L. A., and Loureiro, A. A. F. (2025). A novel prediction technique for federated learning. *IEEE Transactions on Emerging Topics in Computing*, 13(1):5–21.

[da República 2018] da República, P. (2018). Lei geral de proteção de dados. `https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm`. [Accessed 07-04-2025].

[Damoun et al. 2025] Damoun, F., Seba, H., and State, R. (2025). Federated learning-based tokenizer for domain-specific language models in finance. In Aiello, L. M., Chakraborty, T., and Gaito, S., editors, *Social Networks Analysis and Mining*, pages 3–20, Cham. Springer Nature Switzerland.

[de Souza et al. 2024] de Souza, A. M., Maciel, F., da Costa, J. B., Bittencourt, L. F., Cerqueira, E., Loureiro, A. A., and Villas, L. A. (2024). Adaptive client selection with personalization for communication efficient federated learning. *Ad Hoc Networks*, 157:103462.

[Dettmers et al. 2023] Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. (2023). Qlora: Efficient finetuning of quantized llms.

[Devlin et al. 2019] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding.

[Faceli et al. 2021] Faceli, K., Lorena, A. C., Gama, J., Almeida, T. A. d., and Carvalho, A. C. P. d. L. F. d. (2021). *Inteligência artificial: uma abordagem de aprendizado de máquina*. LTC.

[Filho et al. 2022] Filho, R. R., Alberts, E., Gerostathopoulos, I., Porter, B., and Costa, F. M. (2022). Emergent web server: An exemplar to explore online learning in compositional self-adaptive systems. In *2022 International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pages 36–42.

[Friha et al. 2024] Friha, O., Amine Ferrag, M., Kantarci, B., Cakmak, B., Ozgun, A., and Ghoualmi-Zine, N. (2024). Llm-based edge intelligence: A comprehensive survey on architectures, applications, security and trustworthiness. *IEEE Open Journal of the Communications Society*, 5:5799–5856.

[Fu et al. 2023] Fu, L., Zhang, H., Gao, G., Zhang, M., and Liu, X. (2023). Client selection in federated learning: Principles, challenges, and opportunities. *IEEE Internet of Things Journal*, 10(24):21811–21819.

[Google 2023] Google (2023). Organize your Google Photos library with these two updates — blog.google. `https://blog.google/products/photos/google-photos-organization-updates-november-2023/`. [Accessed 07-04-2025].

[Guo et al. 2025] Guo, S., Wang, Y., Feng, B., and Feng, C. (2025). *Large Language Modelx2010;Assisted Semantic Communication Systems*, pages 163–182.

[Guo et al. 2023] Guo, T., Guo, S., and Wang, J. (2023). pfedprompt: Learning personalized prompt for vision-language models in federated learning. In *Proceedings of the ACM Web Conference 2023*, WWW '23, page 1364–1374, New York, NY, USA. Association for Computing Machinery.

[Han et al. 2024] Han, Z., Gao, C., Liu, J., Zhang, J., and Zhang, S. Q. (2024). Parameter-efficient fine-tuning for large models: A comprehensive survey.

[Hu et al. 2021] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. (2021). Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

[Jarczewski et al. 2024] Jarczewski, R. O., Cerqueira, E., Bittencourt, L. F., Loureiro, A. A. F., Villas, L. A., and de Souza, A. M. (2024). Let's federate - effective communication strategy for dynamic client participation. In *2024 International Conference on Machine Learning and Applications (ICMLA)*, pages 361–368.

[Jia et al. 2025] Jia, N., Qu, Z., Ye, B., Wang, Y., Hu, S., and Guo, S. (2025). A comprehensive survey on communication-efficient federated learning in mobile edge environments. *IEEE Communications Surveys & Tutorials*, pages 1–1.

[Kalita 2024] Kalita, A. (2024). Large language models (llms) for semantic communication in edge-based iot networks. *arXiv preprint arXiv:2407.20970*.

[Kuang et al. 2024] Kuang, W., Qian, B., Li, Z., Chen, D., Gao, D., Pan, X., Xie, Y., Li, Y., Ding, B., and Zhou, J. (2024). Federatedscope-llm: A comprehensive package for fine-tuning large language models in federated learning. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 5260–5271.

[Lan et al. 2023] Lan, G., Liu, X.-Y., Zhang, Y., and Wang, X. (2023). Communication-efficient federated learning for resource-constrained edge devices. *IEEE Transactions on Machine Learning in Communications and Networking*, 1:210–224.

[Li et al. 2024] Li, J., Sangalay, A., Cheng, C., Tian, Y., and Yang, J. (2024). Fine tuning large language model for secure code generation. In *Proceedings of the 2024 IEEE/ACM First International Conference on AI Foundation Models and Software Engineering*, FORGE '24, page 86–90, New York, NY, USA. Association for Computing Machinery.

[Li et al. 2025] Li, Q., Lyu, S., and Wen, J. (2025). Optimal client selection of federated learning based on compressed sensing. *IEEE Transactions on Information Forensics and Security*.

[Lira et al. 2024] Lira, O. G., Caicedo, O. M., and Fonseca, N. L. S. d. (2024). Large language models for zero touch network configuration management. *IEEE Communications Magazine*, pages 1–8.

[Llama-Team 2024] Llama-Team (2024). The llama 3 herd of models.

[Lo et al. 2021] Lo, S. K., Lu, Q., Paik, H.-Y., and Zhu, L. (2021). Flra: A reference architecture for federated learning systems. In Biffl, S., Navarro, E., Löwe, W., Sirjani, M., Mirandola, R., and Weyns, D., editors, *Software Architecture*, pages 83–98, Cham. Springer International Publishing.

[Lo et al. 2022] Lo, S. K., Lu, Q., Zhu, L., Paik, H.-Y., Xu, X., and Wang, C. (2022). Architectural patterns for the design of federated learning systems. *J. Syst. Softw.*, 191(C).

[Lu et al. 2024] Lu, Z., Pan, H., Dai, Y., Si, X., and Zhang, Y. (2024). Federated learning with non-iid data: A survey. *IEEE Internet of Things Journal*, 11(11):19188–19209.

[Luo et al. 2024] Luo, W., Keung, J. W., Yang, B., Ye, H., Goues, C. L., Bissyandé, T. F., Tian, H., and Le, B. (2024). When fine-tuning llms meets data privacy: An empirical study of federated learning in llm-based program repair.

[Maciel et al. 2024] Maciel, F., de Souza, A. M., Bittencourt, L. F., Villas, L. A., and Braun, T. (2024). Federated learning energy saving through client selection. *Pervasive and Mobile Computing*, 103:101948.

[Mani et al. 2023] Mani, S. K., Zhou, Y., Hsieh, K., Segarra, S., Eberl, T., Azulai, E., Frizler, I., Chandra, R., and Kandula, S. (2023). Enhancing network management using code generated by large language models. HotNets '23, page 196–204, New York, NY, USA. Association for Computing Machinery.

[Martins et al. 2024] Martins, B. S., Souza, A. M., Rosário, D., Astudillo, C. A., Cerqueira, E., and Villas, L. (2024). Partial training mechanism to handle the impact of stragglers in federated learning with heterogeneous clients. In *2024 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–6.

[McMahan et al. 2017] McMahan, B., Moore, E., Ramage, D., Hampson, S., and Arcas, B. A. y. (2017). Communication-Efficient Learning of Deep Networks from Decentralized Data. In Singh, A. and Zhu, J., editors, *Proceedings of the 20th International*

Conference on Artificial Intelligence and Statistics, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR.

[Mikolov et al. 2013] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space.

[Nanayakkara et al. 2024] Nanayakkara, S. I., Pokhrel, S. R., and Li, G. (2024). Understanding global aggregation and optimization of federated learning. *Future Generation Computer Systems*, 159:114–133.

[Nguyen et al. 2024] Nguyen, L. X., Le, H. Q., Tun, Y. L., Aung, P. S., Tun, Y. K., Han, Z., and Hong, C. S. (2024). An efficient federated learning framework for training semantic communication systems. *IEEE Transactions on Vehicular Technology*, 73(10):15872–15877.

[OpenAI 2024] OpenAI (2024). Gpt-4 technical report.

[Penedo et al. 2024] Penedo, G., Kydlíček, H., allal, L. B., Lozhkov, A., Mitchell, M., Raffel, C., Werra, L. V., and Wolf, T. (2024). The fineweb datasets: Decanting the web for the finest text data at scale.

[Peng et al. 2024] Peng, L., Luo, G., Zhou, S., Chen, J., Xu, Z., Sun, J., and Zhang, R. (2024). An in-depth evaluation of federated learning on biomedical natural language processing for information extraction. *npj Digital Medicine*, 7(1):127.

[Piccialli et al. 2025] Piccialli, F., Chiaro, D., Qi, P., Bellandi, V., and Damiani, E. (2025). Federated and edge learning for large language models. *Information Fusion*, 117:102840.

[Puppala et al. 2024] Puppala, S., Hossain, I., Alam, M. J., and Talukder, S. (2024). Scan: A healthcare personalized chatbot with federated learning based gpt. In *2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 1945–1951.

[Qiyang Zhao 2024] Qiyang Zhao, Hang Zou, M. B. M. D. (2024). Semantic communications. In Mohammad A. Matin, Sotirios K. Goudos, G. K. K., editor, *Artificial Intelligence for Future Networks*, pages 131–149. Wiley.

[Radford et al. 2019] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

[Riedel et al. 2024] Riedel, P., Schick, L., von Schwerin, R., Reichert, M., Schaudt, D., and Hafner, A. (2024). Comparative analysis of open-source federated learning frameworks - a literature-based survey and review. *International Journal of Machine Learning and Cybernetics*, 15(11):5257–5278.

[Sarwar 2024] Sarwar, S. M. (2024). Fedmentalcare: Towards privacy-preserving fine-tuned llms to analyze mental health status using federated learning framework.

[Sennrich et al. 2016] Sennrich, R., Haddow, B., and Birch, A. (2016). Neural machine translation of rare words with subword units.

[Signal 2013] Signal (2013). Signal Messenger: Speak Freely — signal.org. `https://signal.org/pt_BR/`. [Accessed 07-04-2025].

[Singh and Adhikari 2025] Singh, N. and Adhikari, M. (2025). Selffed: Self-adaptive federated learning with non-iid data on heterogeneous edge devices for bias mitigation and enhance training efficiency. *Information Fusion*, 118:102932.

[Smith et al. 2017] Smith, V., Chiang, C.-K., Sanjabi, M., and Talwalkar, A. (2017). Federated multi-task learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 4427–4437, Red Hook, NY, USA. Curran Associates Inc.

[Soltani et al. 2023] Soltani, B., Zhou, Y., Haghighi, V., and Lui, J. C. S. (2023). A survey of federated evaluation in federated learning. In Elkind, E., editor, *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, pages 6769–6777. International Joint Conferences on Artificial Intelligence Organization. Survey Track.

[Souza et al. 2023] 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.

[Sun et al. 2025] Sun, H., Tian, H., Ni, W., Zheng, J., Niyato, D., and Zhang, P. (2025). Federated low-rank adaptation for large models fine-tuning over wireless networks. *IEEE Transactions on Wireless Communications*, 24(1):659–675.

[Talasso et al. 2024] 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.

[Tam et al. 2024] Tam, K., Tian, C., Li, L., Zhao, H., and Xu, C. (2024). Fedhybrid: Breaking the memory wall of federated learning via hybrid tensor management. In *Proceedings of the 22nd ACM Conference on Embedded Networked Sensor Systems*, pages 394–408.

[Tang and Deng 2024] Tang, Y. and Deng, Y. (2024). Current research and prospects of federated language large models in the medical field. In Piccaluga, P. P. and Baloch, Z., editors, *Third International Conference on Biomedical and Intelligent Systems (IC-BIS 2024)*, volume 13208, page 1320838. International Society for Optics and Photonics, SPIE.

[Tao et al. 2025] Tao, Y., Yang, R., Zeng, K., Yin, C., Lu, Y., Lu, W., Shi, Y., Wang, B., and Huang, B. (2025). Flft: A large-scale pre-training model distributed fine-tuning method that integrates federated learning strategies. *IEEE Access*.

[Van Nguyen et al. 2024] Van Nguyen, C., Shen, X., Aponte, R., Xia, Y., Basu, S., Hu, Z., Chen, J., Parmar, M., Kunapuli, S., Barrow, J., et al. (2024). A survey of small language models. *arXiv preprint arXiv:2410.20011*.

[Vaswani 2017] Vaswani, A. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*.

[Wang et al. 2024] Wang, D., Kim, D., Jin, B., Zhao, X., Fu, T., Yang, S., and Liu, X.-Y. (2024). Finlora: Finetuning quantized financial large language models using low-rank adaptation.

[Weyssow et al. 2025] Weyssow, M., Zhou, X., Kim, K., Lo, D., and Sahraoui, H. (2025). Exploring parameter-efficient fine-tuning techniques for code generation with large language models. *ACM Trans. Softw. Eng. Methodol.* Just Accepted.

[Woisetschläger et al. 2024] Woisetschläger, H., Erben, A., Wang, S., Mayer, R., and Jacobsen, H.-A. (2024). Federated fine-tuning of llms on the very edge: The good, the bad, the ugly. In *Proceedings of the Eighth Workshop on Data Management for End-to-End Machine Learning*, DEEM '24, page 39–50, New York, NY, USA. Association for Computing Machinery.

[Wu et al. 2024a] Wu, F., Li, Z., Li, Y., Ding, B., and Gao, J. (2024a). Fedbiot: Llm local fine-tuning in federated learning without full model. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3345–3355.

[Wu et al. 2025a] Wu, F., Liu, X., Wang, H., Wang, X., Su, L., and Gao, J. (2025a). Towards federated RLHF with aggregated client preference for LLMs. In *The Thirteenth International Conference on Learning Representations*.

[Wu et al. 2024b] Wu, J., Dong, F., Leung, H., Zhu, Z., Zhou, J., and Drew, S. (2024b). Topology-aware federated learning in edge computing: A comprehensive survey. *ACM Comput. Surv.*, 56(10).

[Wu et al. 2025b] Wu, Y., Tian, C., Li, J., Sun, H., Tam, K., Li, L., and Xu, C. (2025b). A survey on federated fine-tuning of large language models. *arXiv preprint arXiv:2503.12016*.

[Xie et al. 2023] Xie, Y., Wang, Z., Gao, D., Chen, D., Yao, L., Kuang, W., Li, Y., Ding, B., and Zhou, J. (2023). Federatedscope: A flexible federated learning platform for heterogeneity. *Proceedings of the VLDB Endowment*, 16(5):1059–1072.

[Yan et al. 2025] Yan, N., Su, Y., Deng, Y., and Schober, R. (2025). Federated fine-tuning of llms: Framework comparison and research directions. *arXiv preprint arXiv:2501.04436*.

[Yang et al. 2023] Yang, F.-E., Wang, C.-Y., and Wang, Y.-C. F. (2023). Efficient Model Personalization in Federated Learning via Client-Specific Prompt Generation . In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 19102–19111, Los Alamitos, CA, USA. IEEE Computer Society.

[Ye et al. 2023] Ye, M., Fang, X., Du, B., Yuen, P. C., and Tao, D. (2023). Heterogeneous federated learning: State-of-the-art and research challenges. *ACM Comput. Surv.*, 56(3).

[Zeng et al. 2024] Zeng, J., Chen, B., Deng, Y., Chen, W., Mao, Y., and Li, J. (2024). Fine-tuning of financial large language model and application at edge device. In *Proceedings of the 3rd International Conference on Computer, Artificial Intelligence and Control Engineering*, CAICE '24, page 42–47, New York, NY, USA. Association for Computing Machinery.

[Zhang and You 2024] Zhang, Y. and You, Y. (2024). Speedloader: An i/o efficient scheme for heterogeneous and distributed llm operation. *Advances in Neural Information Processing Systems*, 37:34637–34655.

[Zhao et al. 2023] Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., et al. (2023). A survey of large language models. *arXiv preprint arXiv:2303.18223*.

[Zhao et al. 2024] Zhao, Y., Yue, Y., Hou, S., Cheng, B., and Huang, Y. (2024). Lamosc: Large language model-driven semantic communication system for visual transmission. *IEEE Transactions on Cognitive Communications and Networking*, 10(6):2005–2018.

[Zhu et al. 2021] Zhu, H., Xu, J., Liu, S., and Jin, Y. (2021). Federated learning on non-iid data: A survey. *Neurocomputing*, 465:371–390.

[Zhuang et al. 2022] Zhuang, W., Gan, X., Wen, Y., and Zhang, S. (2022). Easyfl: A low-code federated learning platform for dummies. *IEEE Internet of Things Journal*.

