Ajuste fino de modelos Transformers através de técnicas PEFT (Parameter Efficient Fine-Tuning)

A personalização de modelos de linguagem para tarefas e/ou comportamentos específicos por meio do fine-tuning, tem ganhado relevância no universo de Inteligência Artificial Generativa (IAG). Neste estudo, exploramos como o ajuste fino de modelos de linguagem pode transformar tarefas específicas em empresas e atender às demandas personalizadas de clientes.

Objetivo

Este estudo teve como objetivo principal documentar e analisar o processo de fine-tuning de modelos de linguagem. Nosso foco foi entender como essas técnicas podem ajudar a equipe de IAG a desenvolver modelos personalizados que atendam às necessidades específicas de cada cliente.

Estrutura

O estudo foi organizado em três etapas:

1. **Introdução Teórica**: Definimos o que é o fine-tuning, suas vantagens, desvantagens e limitações, e abordamos a arquitetura Transformer.
2. **Técnicas de Fine-Tuning Eficiente (PEFT)**: Exploramos técnicas eficientes de fine-tuning, com destaque para LoRA e QLoRA.
3. **Experimentação**: Realizamos implementações práticas, focando na tarefa de sumarização de conversas para demonstrar a eficácia das técnicas.

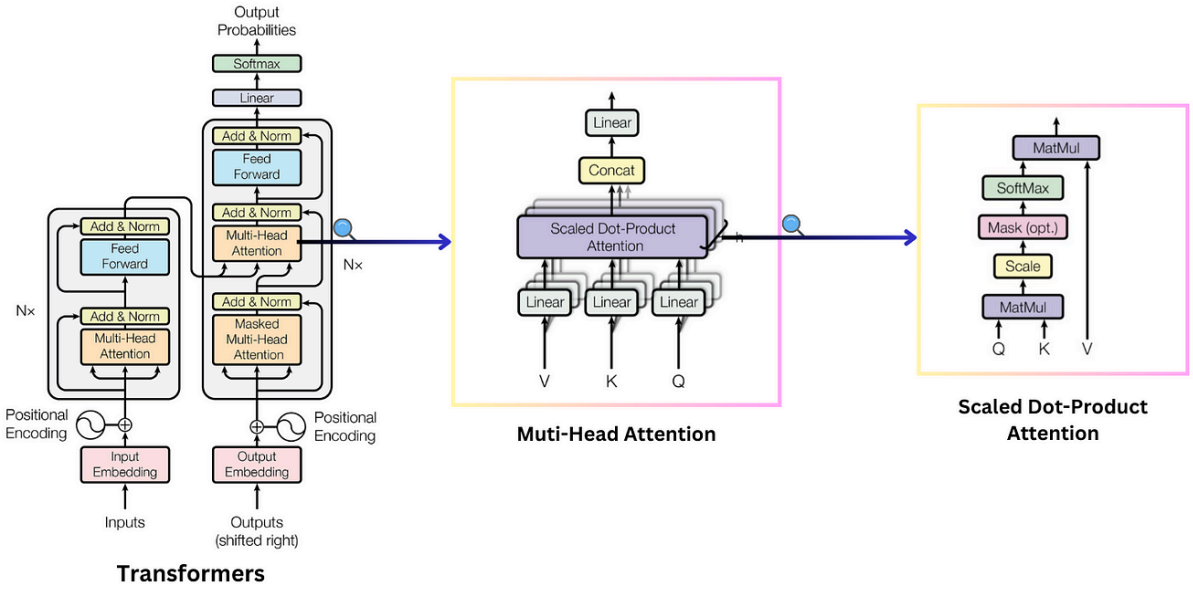
Introdução Teórica

Arquitetura Transformers

A arquitetura Transformer, introduzida em 2017 no artigo "Attention is All You Need", revolucionou o campo de processamento de linguagem natural (NLP) e aprendizado profundo. Diferente das arquiteturas recorrentes e convolucionais, os Transformers se baseiam inteiramente no mecanismo de atenção, permitindo um paralelismo mais eficiente e melhor desempenho em tarefas de sequência.

Arquitetura Original

A arquitetura Transformer é composta por dois componentes principais: o Encoder e o Decoder. O Encoder é responsável por mapear uma entrada em uma representação intermediária, enquanto o Decoder usa essa representação intermediária para gerar a saída.



Mecanismo de Atenção

Uma característica chave dos modelos Transformer é que eles são construídos com camadas especiais chamadas camadas de atenção. Essa camada dirá ao modelo para prestar atenção específica a certas palavras na frase que você passou (e mais ou menos ignorar as outras) ao lidar com a representação de cada palavra considerando o seu contexto.

O mecanismo de atenção é central na arquitetura Transformer. Existem três componentes principais na atenção:

- **Query (Q)**: Representa a entrada atual que deseja se atentar a outras partes da sequência.
- **Key (K)** e **Value (V)**: Representam as características da sequência a serem comparadas com a Query.

A atenção é calculada como um produto escalar entre Q e K, seguido por uma normalização via Softmax para obter os pesos de atenção, que são então aplicados aos Values. No contexto

de multi-head attention, essa operação é realizada várias vezes em paralelo, permitindo que o modelo aprenda diferentes padrões de atenção.

Bloco de Encoder

O codificador recebe uma entrada e constrói uma representação dela (seus recursos). Isso significa que o modelo é otimizado para adquirir entendimento da entrada.

Cada bloco de Encoder é composto por dois sub-blocos principais:

- **Multi-Head Self-Attention**: Permite que o modelo se atente a diferentes partes da sequência de entrada simultaneamente. A camada de atenção é composta por vários "heads" que executam a atenção em paralelo, e seus resultados são concatenados e projetados.
- **Feed-Forward Neural Network**: Consiste em duas camadas lineares com uma ativação não-linear (ReLU) no meio. A primeira camada aumenta a dimensionalidade dos dados e a segunda camada reduz de volta à dimensão original.

Ambos os sub-blocos são envolvidos por uma normalização de camada (Layer Normalization) e um mecanismo de conexão residual que ajuda na propagação do gradiente durante o treinamento.

Bloco de Decoder

O decodificador usa a representação do codificador (recursos) junto com outras entradas para gerar uma sequência de destino. Isso significa que o modelo é otimizado para gerar saídas.

O Decoder é semelhante ao Encoder, mas com algumas diferenças adicionais:

- **Masked Multi-Head Self-Attention**: Similar à atenção no Encoder, mas com uma máscara que impede que a posição atual atenda às posições futuras, essencial para a geração sequencial.
- **Multi-Head Attention**: Esta camada realiza atenção sobre a saída do Encoder, permitindo que o Decoder se atente às partes relevantes da entrada.
- **Feed-Forward Neural Network**: Igual à do Encoder.

Assim como no Encoder, todos os sub-blocos do Decoder são envolvidos por normalização de camada e conexões residuais.

Variações de Modelo

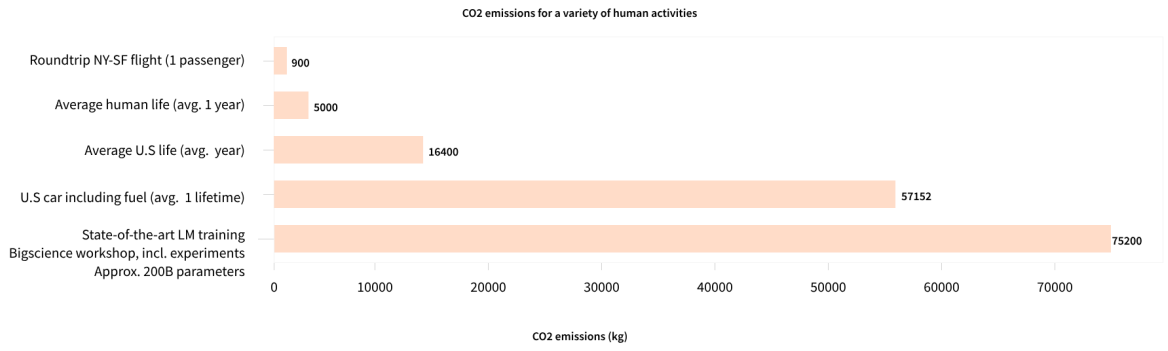
Cada uma das partes da arquitetura podem ser usadas de forma independente, dependendo da tarefa:

- **Modelos somente de codificador**: bom para tarefas que exigem compreensão da entrada, como Classificação de sentença e reconhecimento de entidade nomeada.

Esses modelos geralmente são caracterizados como tendo atenção “bidirecional” e são frequentemente chamados de modelos de codificação automática.
    - BERT
    - DistilBERT
    - RoBERTa
- **Modelos somente decodificadores**: bom para tarefas generativas, como Geração de texto. Em cada etapa, para uma determinada palavra, as camadas de atenção só podem acessar as palavras posicionadas antes dela na frase. Esses modelos geralmente são chamados de modelos auto-regressivos.
    - GPT
    - GPT-2
- **Modelos de codificador-decodificador ou modelos de sequência a sequência**: bom para tarefas generativas que exigem uma entrada, como Sumarização, tradução, perguntas e respostas gerativas
    - BART
    - T5

Ajuste Fino

Transformers são modelos grandes e complexos, e o aumento de seu tamanho e da quantidade de dados pré-treinados é uma estratégia comum para melhorar o desempenho. No entanto, treinar esses modelos do zero é altamente custoso em termos de tempo, recursos computacionais e até mesmo ambientalmente, devido ao consumo de energia. Por isso, compartilhar modelos pré-treinados se tornou essencial, permitindo que a comunidade reduza custos e aproveite os ganhos de eficiência.



O `Transfer Learning` é a técnica que aproveita modelos pré-treinados em uma tarefa para serem adaptados em outra tarefa específica. Nos grandes modelos de linguagem (LLMs), como GPT ou BERT, essa técnica permite que o conhecimento adquirido em grandes corpora seja reutilizado e ajustado para tarefas específicas, como classificação de texto, tradução

automática ou resposta a perguntas. Essa adaptação é feita ajustando o modelo supervisionadamente com novos dados rotulados da tarefa-alvo.

O `Fine-Tuning` (ajuste fino) é o processo de especializar um modelo pré-treinado em uma tarefa específica. Ele ajusta os pesos do modelo com base em um conjunto de dados rotulados menor, mantendo o conhecimento geral adquirido no pré-treinamento. Essa abordagem requer menos dados e recursos, resultando em ganhos de eficiência significativos sem comprometer a performance para a nova tarefa.

Razões para Utilizar Fine-Tuning em Modelos de Linguagem

Existem diversos motivos pelos quais o fine-tuning se mostra vantajoso, especialmente em aplicações corporativas. Dentre eles, destacam-se:

1. **Privacidade de Dados**: Permite criar modelos customizados sem a necessidade de compartilhar dados sensíveis com terceiros.
2. **Comportamento Personalizado**: Ajusta o modelo para responder de forma específica, que o modelo original pode não ser capaz de fazer.
3. **Resolução de Problemas Restritos**: Lida com questões específicas, como a sumarização de mensagens complexas.
4. **Limitações de Engenharia de Prompt e RAG**: Em casos onde outras técnicas de IA não são suficientes, o fine-tuning oferece uma solução robusta e direcionada.

Vantagens do Fine-Tuning

O fine-tuning apresenta várias vantagens importantes para empresas e desenvolvedores:

- **Especificação e Personalização**: Ajusta o modelo de forma a fornecer respostas mais precisas e relevantes.
- **Eficiência de Dados**: Demanda menos dados rotulados comparado ao pré-treinamento.
- **Economia de Recursos Computacionais**: Como o modelo já é pré-treinado, o ajuste fino é mais rápido e exige menos recursos.
- **Autonomia Tecnológica**: Reduz a dependência de fornecedores externos, permitindo o desenvolvimento de modelos próprios.
- **Melhoria na Experiência do Usuário**: Resulta em respostas mais rápidas e precisas, elevando a satisfação do cliente.

Desvantagens e Limitações do Fine-Tuning

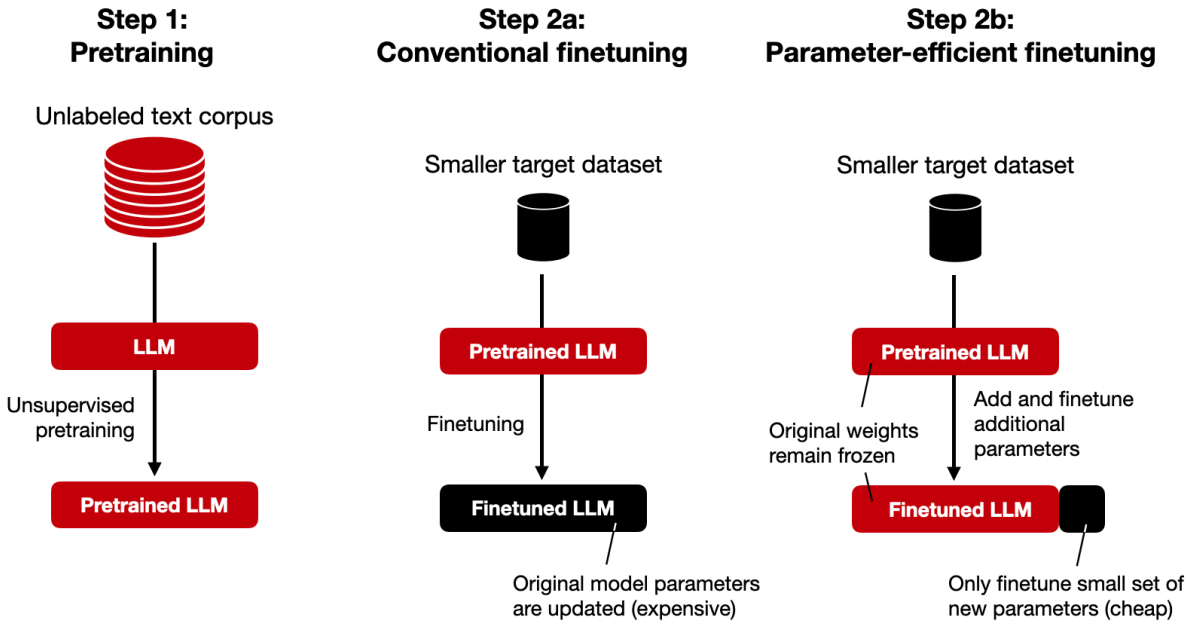
Embora eficaz, o fine-tuning apresenta desafios:

- **Recursos Computacionais Intensivos**: Requer uma quantidade significativa de dados de qualidade e poder computacional.
- **Risco de Overfitting e Esquecimento Catastrófico**: O modelo pode se especializar excessivamente, perdendo a capacidade de generalização.

- **Necessidade de Validação**: A qualidade e o equilíbrio dos dados são críticos para evitar vieses.

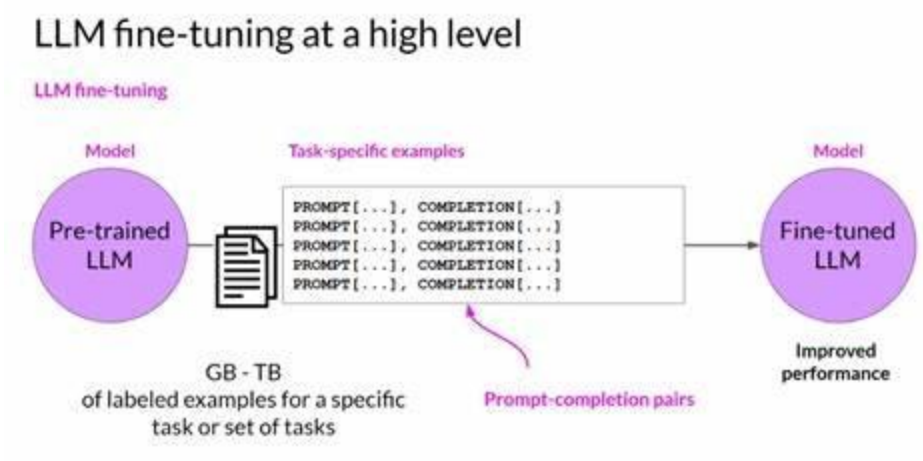
Técnicas de Ajuste Fino

Os processos de ajuste fino podem ter uma visão geral entre 2 categorias, sendo o **Ajuste Total (Full Tuning)** ou **Ajustes Fino de Parâmetros Eficientes (Parameter-Efficient Fine-Tuning, ou PEFT)**



Ajuste Total

O treinamento de todos os parâmetros de um modelo para uma tarefa específica. Uma das técnicas deste tipo de processo é a de Instruct Tuning, que resumidamente considera um conjunto de dados com pares de Prompt-Completion. Podem ser perguntas e respostas, ou instruções e ações tomadas.



Fonte: https://www.coursera.org/learn/generative-ai-with-llms/

O processo de ajuste total tem alta eficiência em especializar os LLMs. No entanto, o fato de o ajuste acontecer com todos os parâmetros do modelo torna esse processo lento e custoso. A alternativa desenvolvida foram os processos de PEFT.

Ajustes Fino de Parâmetros Eficientes (PEFT)

Os métodos de Fine-Tuning Eficiente de Parâmetros (PEFT) podem ser classificados de acordo com dois aspectos principais: sua estrutura conceitual (por exemplo, introdução de novos parâmetros ou ajuste de parâmetros existentes) e seu objetivo primário (minimização do footprint de memória, eficiência de armazenamento ou redução dos custos computacionais). Esses métodos se dividem em três grandes categorias:

Métodos Aditivos

Os métodos aditivos introduzem novos parâmetros no modelo base, geralmente através de camadas adaptadoras pequenas ou ajustando uma parte dos embeddings de entrada (conhecidos como soft prompts). Estes métodos são amplamente utilizados e incluem:

- **Adaptadores**: Pequenas redes densas (totalmente conectadas) que são inseridas após subcamadas específicas dos transformadores, permitindo a adaptação a novas tarefas sem a necessidade de treinar todos os parâmetros do modelo.
- **Soft Prompts**: Ajustes finos aplicados diretamente nos embeddings de entrada do modelo, facilitando sua adaptação às tarefas-alvo sem modificar os parâmetros internos do modelo.

Esses métodos geralmente são eficientes em termos de memória, pois reduzem o tamanho dos gradientes e dos estados do otimizador.

Métodos Seletivos

Os métodos seletivos ajustam apenas uma fração dos parâmetros já existentes no modelo. Isso pode ser feito de diferentes maneiras, como:

- **Ajuste de Camadas Superiores**: Foco no ajuste apenas das camadas superiores da rede, deixando as camadas inferiores intactas.
- **Ajuste de Parâmetros Específicos**: Treinamento seletivo de certos tipos de parâmetros, como vieses, enquanto outros parâmetros permanecem congelados.
- **Atualizações Esparsas**: Seleção de um subconjunto específico de parâmetros para serem treinados. Embora promissora, essa abordagem pode ser computacionalmente mais cara devido à necessidade de identificar os parâmetros mais relevantes.

Apesar da economia em termos de parâmetros treinados, métodos seletivos podem apresentar altos custos computacionais, especialmente em configurações esparsas.

Métodos Baseados em Reparametrização

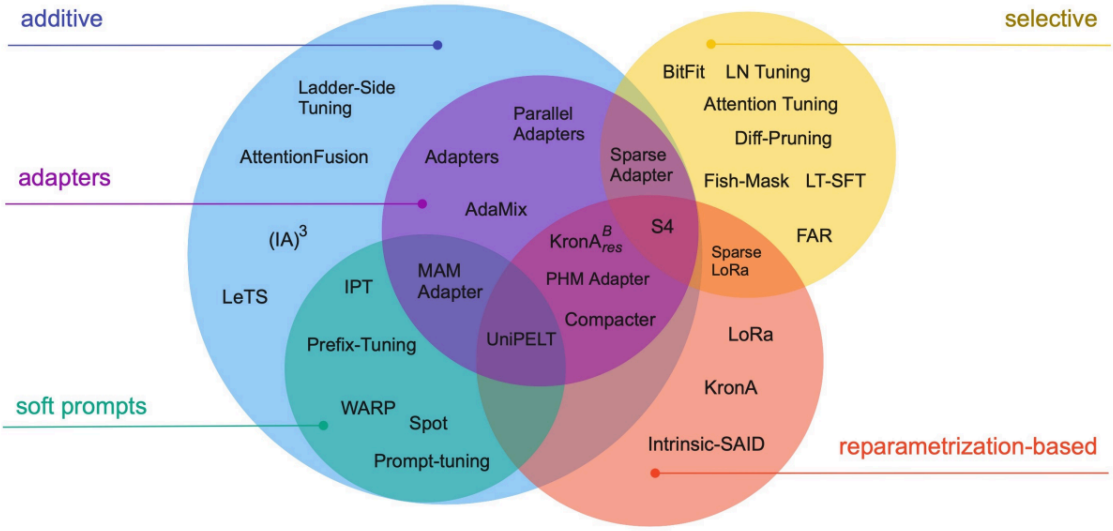
Os métodos baseados em reparametrização reduzem o número de parâmetros treináveis ao utilizar representações de baixa classificação, explorando a redundância presente nas redes neurais. Alguns dos principais métodos incluem:

- **LoRa (Adaptação de Baixa Classificação)**: Utiliza decomposição de matrizes de baixa classificação para representar as atualizações de peso, resultando em uma forma eficiente de ajuste fino.
- **Intrinsic SAID**: Emprega a transformação Fastfood, uma técnica para representar atualizações de baixa classificação de maneira computacionalmente eficiente.

Estes métodos oferecem uma significativa redução no número de parâmetros a serem treinados, tornando-os ideais para situações em que a eficiência de armazenamento e o tempo de treinamento são críticos.

Pontos Adicionais

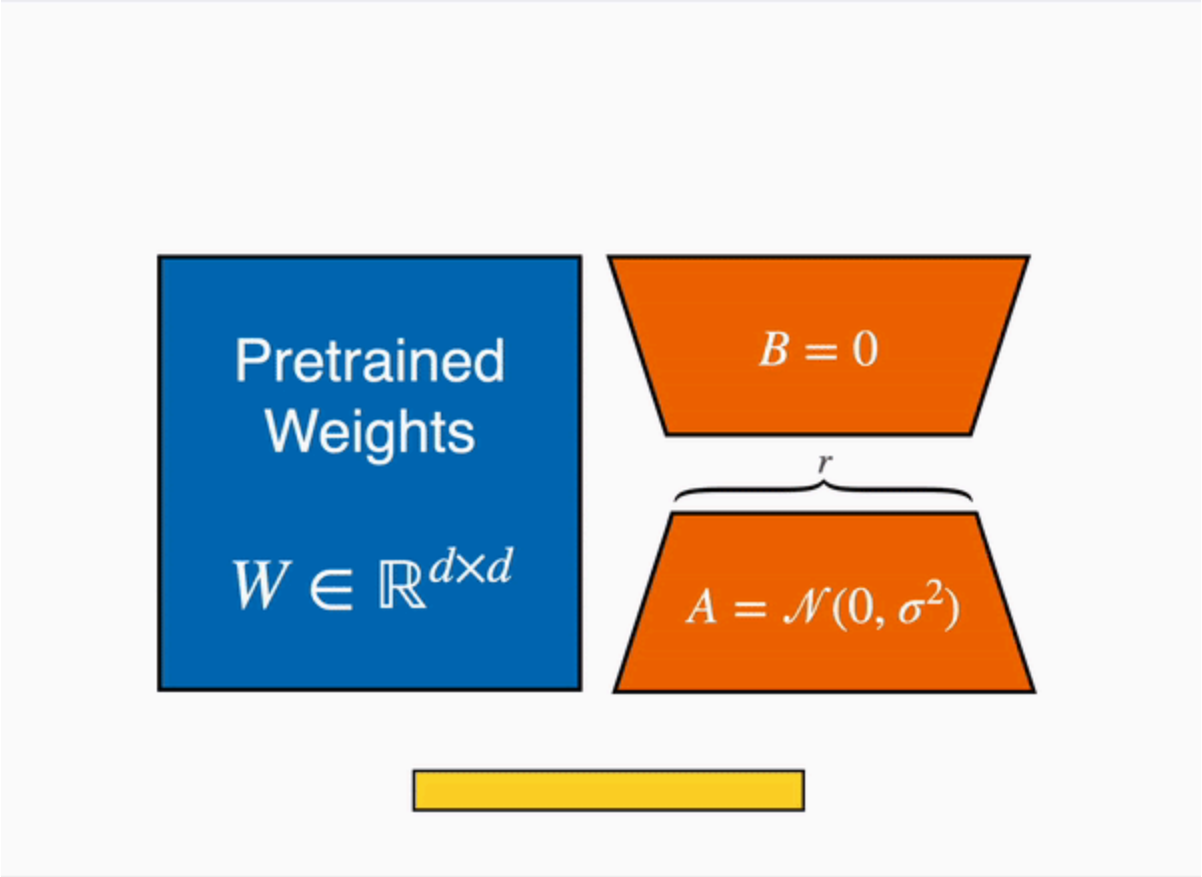
- **Métodos Aditivos**: Embora introduzem novos parâmetros, podem ser mais eficientes em termos de memória geral, reduzindo a quantidade de gradientes e estados do otimizador que precisam ser armazenados.
- **Métodos Seletivos**: Embora promissores para reduzir o número de parâmetros treinados, podem ser computacionalmente intensivos, particularmente em casos de atualizações esparsas.
- **Métodos Híbridos**: Combinações de ideias de diferentes categorias são frequentemente exploradas para maximizar o desempenho, aproveitando o melhor de cada abordagem.



Fonte: [Scaling Down to Scale Up: A Guide to Parameter-Efficient Fine-Tuning" e no módulo PEFT]

LoRA

É um dos métodos PEFT mais populares e um bom ponto de partida para começar no meio do fine tuning eficiente.



LoRA decompõe as atualizações de peso para uma matriz ΔW em um produto de duas matrizes de baixa classificação, Wa e Wb, onde ΔW = WA x Wb.

A matriz de peso original permanece congelada, e apenas as matrizes menores são treinadas, que podem então ser mescladas com os pesos originais para produzir a saída final.

Na prática, é frequentemente aplicado aos blocos de atenção dos modelos Transformer, visando especificamente as matrizes de projeção Wk e Wv em módulos multi-head attention.

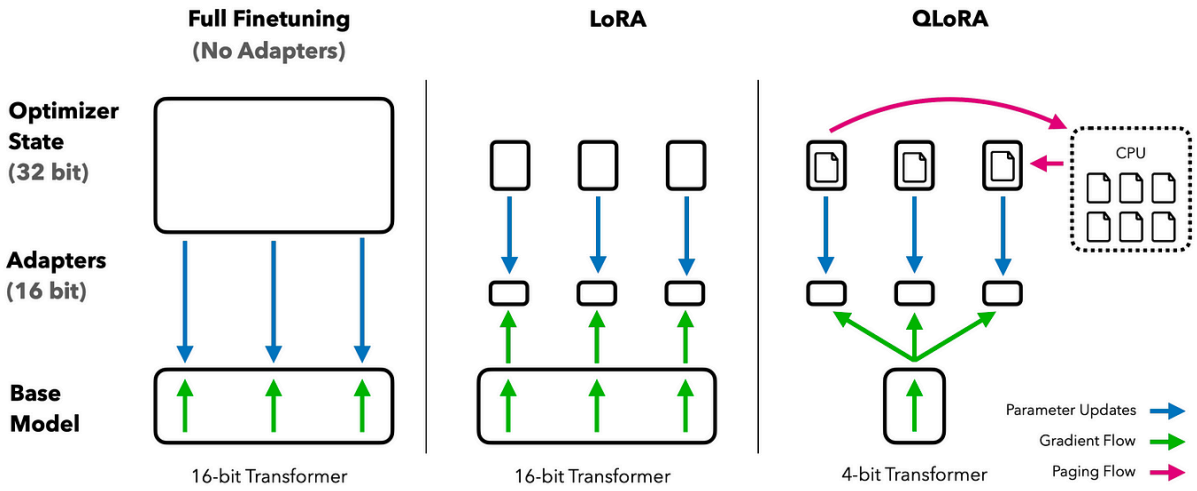
Esta abordagem tem uma série de vantagens:

- Torna o ajuste fino mais eficiente reduzindo drasticamente o número de parâmetros treináveis.
- Os pesos pré-treinados originais são mantidos congelados, o que significa que você pode ter vários modelos LoRA leves e portáteis para várias tarefas posteriores construídas sobre eles.
- É ortogonal a outros métodos eficientes em termos de parâmetros e pode ser combinado com muitos deles.
- O desempenho comparável ao desempenho de modelos totalmente ajustados

QLoRA

QLoRA (Quantized Low-Rank Adaptation), combina a técnica tradicional de quantização com o LoRA para melhorar a eficiência do ajuste fino.

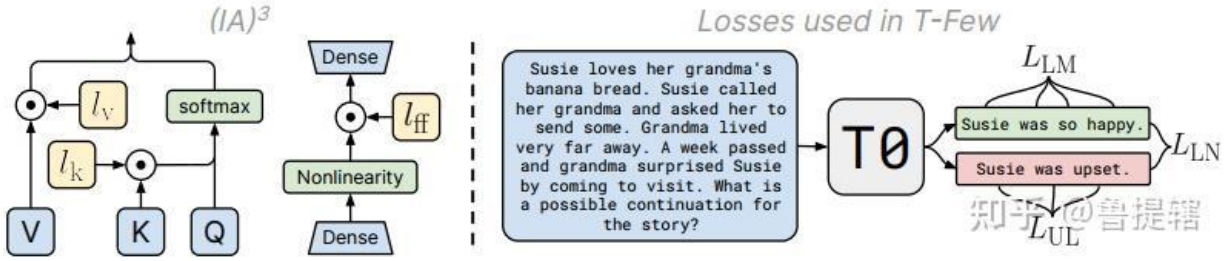
Foi introduzido por Dettmers et al. em maio de 2023 (QLoRA Paper), e eles demonstraram que o QLoRA poderia ajustar um modelo de 65B de parâmetros com uma única GPU de 48GB, preservando o desempenho completo da tarefa de ajuste fino de 16 bits.



Ele apresenta algumas inovações sem reduzir o desempenho:

- 4-bit Normal Float (NF4), um novo tipo de dados que é uma informação teoricamente ideal para pesos normalmente distribuídos, a qual ocupa metade de um byte por elemento.
- Quantização dupla para reduzir o consumo médio de memória quantizando as constantes de quantização, utiliza a implementação bitsandbytes do Hugging Face 's.
- Otimizadores paginados para gerenciar picos de memória.

IA3



(IA)³ (Infused Adapter by Inhibiting and Amplifying Inner Activations) modifica apenas vetores específicos aprendidos associados às camadas de chave, valor e feedforward dentro dos blocos do transformador.

O método introduz três vetores, `lv`, `lk` e `lf`, que reescalam ativações nas camadas de atenção e feedforward. Essa abordagem mantém a maioria dos pesos do modelo congelados, reduzindo drasticamente o número de parâmetros treináveis.

Experimentação: Ajuste de um LLM para sumarização de diálogos entre cliente e agente

A experimentação prática é essencial para validar a eficiência das técnicas de fine-tuning e demonstrar seu impacto em tarefas específicas, como a sumarização de textos. Neste estudo, aplicamos e analisamos as técnicas **LoRA**, **QLoRA** e **IA3**, explorando sua eficácia em modelos de linguagem grandes (LLMs) e menores, especialmente em contextos com limitação de hardware.

Essa experimentação envolve a prática de métodos de fine tuning em modelos de linguagem para a tarefa de sumarização de conversas de atendimento, passando por todo ciclo de aprendizado de máquina, desde a preparação dos dados ao ajuste de hiperparâmetros e avaliação.

Objetivo e Metodologia

O objetivo principal desta etapa foi testar o conhecimento teórico adquirido, validando a eficiência das técnicas de fine-tuning com foco em aplicações de sumarização. A implementação incluiu:

1. **Pesquisa e comparação de frameworks**: Avaliamos frameworks como *axoltl*, *torchtune*, *llama-tuner* e *peft* para identificar as ferramentas mais adequadas para o ajuste fino.
2. **Notebook de Fine-Tuning para Sumarização**: Criamos um notebook documentado que cobre o processo completo de fine-tuning para sumarização de conversas.
   PEFT_Finetuning_T5_Base.ipynb
3. **Instrumentação de Experimentos com Weights and Biases (W&B)**: Usamos a ferramenta Sweep do *W&B* para busca de hiperparâmetros e análise do treinamento em tempo real.

Seleção de Ferramentas e Frameworks

Optamos pela biblioteca *peft*, integrada ao módulo transformers do Hugging Face. Ela permite mais controle sobre cada etapa do ciclo de treinamento, proporcionando uma visão detalhada do processo de fine-tuning. Além disso, ela apresenta a maior gama de métodos eficientes,

uma documentação teórica e referências extensas, uma grande comunidade, desenvolvimento ativo, além de muitos materiais publicados que a utilizam.

O Weights and Biases (W&B) é uma ferramenta de instrumentação para treinamento de modelos de machine learning, que permite rastrear experimentos, visualizar métricas em tempo real, registrar hiperparâmetros e resultados, e colaborar com outros membros da equipe. Ele é integrado com bibliotecas populares, como PyTorch, TensorFlow e Hugging Face, facilitando a instrumentação automática de modelos e a análise dos dados de treinamento.

A funcionalidade do W&B Sweeps permite realizar buscas automatizadas de hiperparâmetros, testando diferentes combinações para otimizar o desempenho do modelo. Você pode configurar os parâmetros de sweep com estratégias de busca como grid search, random search ou bayesian optimization. O W&B coordena os experimentos, registra os resultados e oferece comparações visuais entre diferentes execuções, ajudando a identificar os melhores conjuntos de hiperparâmetros de forma eficiente.

Dataset

Para este estudo, selecionamos o TweetSumm, um dataset especializado em diálogos de suporte ao cliente provenientes de interações reais. O conjunto contém 1.100 diálogos reconstruídos a partir do dataset "Customer Support On Twitter" do Kaggle, enriquecidos com três resumos extrativos e três abstrativos por diálogo, todos gerados por anotadores humanos.

Utilizamos a versão formatada disponível no Hugging Face Datasets (Andyrasika/TweetSumm-tuned), que organiza os dados em três campos principais: 'conversation' (texto da conversa), 'summary' (resumo humano) e 'text' (prompt formatado com instruções). O dataset está dividido em 879 exemplos para treino, 110 para validação e 110 para teste.

Eis um exemplo representativo do dataset:

```
Unset
{
    "conversation": "user: Do Apple updates always delete your recent contacts? I've easily lost over 50 contacts in a day. Thanks Oh and my battery sucks after the update as usual too. I'm sure that has nothing to do with the update, though. Nothing at all. agent: We can help with your missing contacts and battery. Has the device been restarted? Do you use iCloud for syncing contacts? user: Yes and yes. agent: Have you tried navigating to Settings > [your name] > iCloud and checking to ensure the contacts toggle is on? user: Yes. agent: Thanks for the screenshot and checking. Please DM us and we'll further review this for you.",
```

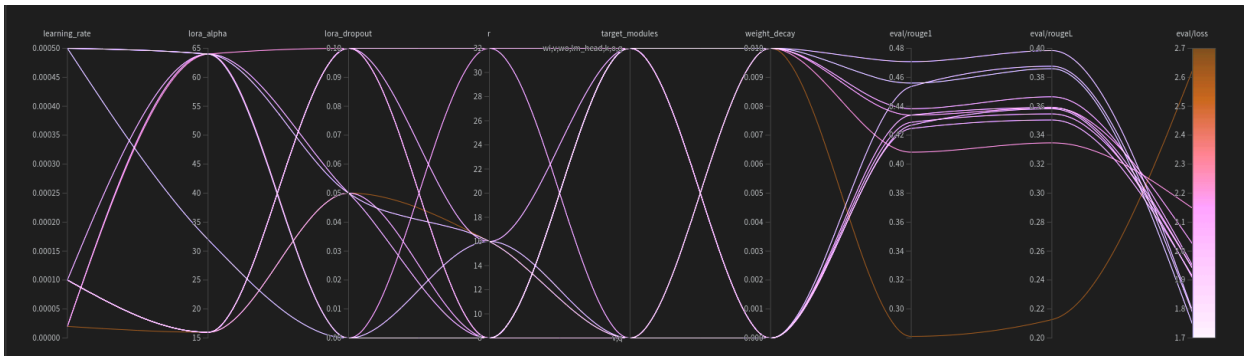
```
    "summary": "The customer is complaining that they have lost more than 50
contacts and battery also sucks after the update. The agent asked to dm them
for further review on that issue."
}
```

Hyper Tuning

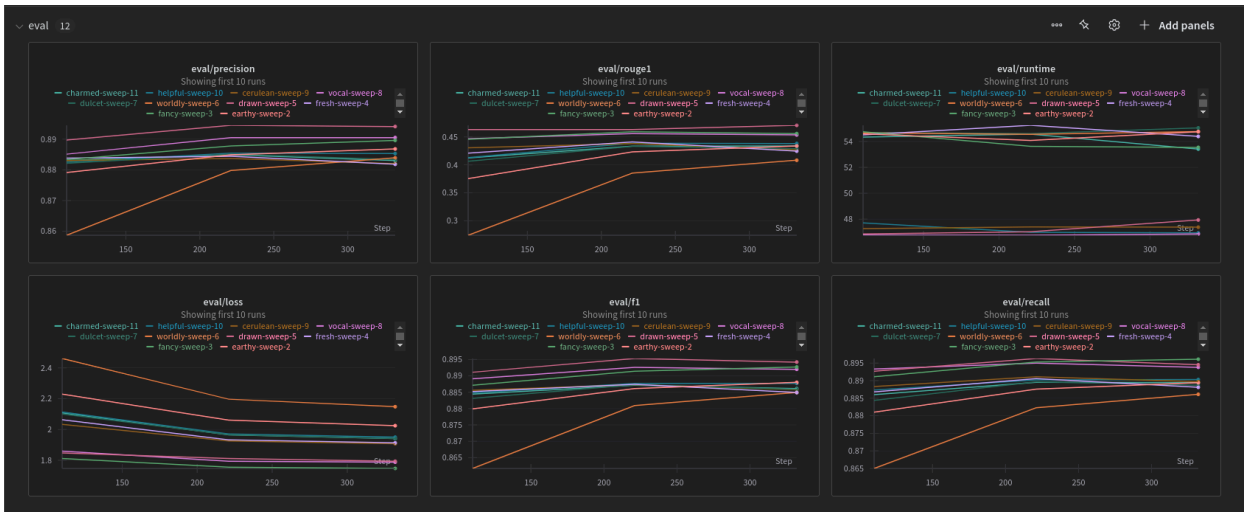
Utilizando a ferramenta Weights & Biases, realizamos uma busca de hiperparâmetros com 11 combinações diferentes para cada técnica. Assim obtemos os melhores parâmetros para determinado modelo e técnica.

A ferramenta Weights & Biases (W&B) foi fundamental para monitorar e otimizar o treinamento dos modelos, principalmente ao realizar um Sweep para buscar a melhor combinação de hiperparâmetros. Essas varreduras (Sweeps) permitiram observar como diferentes configurações de parâmetros impactaram a qualidade dos resultados, facilitando ajustes rápidos e coerentes.

As curvas geradas pelo W&B em tempo real forneceram insights essenciais sobre o comportamento do modelo durante o treinamento e ajudaram a refinar as configurações de hiperparâmetros. Isso tornou o processo mais estruturado e eficiente, evitando ajustes aleatórios e reduzindo desperdícios de recursos computacionais.



Essa figura demonstra o que é chamado de Sweep do W&B, curvas que demonstram a combinação de parâmetros utilizados em cada corrida/treino e podemos associá-las a métricas, dados de referência. Assim traçando o comportamento em que a configuração de parâmetros do modelo pode afetar.



As outras figuras exibem as curvas de diferentes métricas acompanhadas durante o treinamento.

Estes resultados correspondem à busca de hiperparâmetros que fiz com o modelo T5-base utilizando a técnica QLoRA, totalizando 11 treinos com diferentes combinações. Essa busca foio replicada para as técnicas LoRA e IA3.

Aqui está a configuração usada e o acesso ao registro das experimentações:

- LoRA: [T5 - base LoRA Fietune Sweep]

```
Python
lora_sweep_configuration = {
```

```
    'method': 'random',
    'parameters': {
        'learning_rate': {'values': [5e-4, 1e-4, 1e-3]},
        'weight_decay' : {'values': [0.01, 0.0]},
        'r': {'values': [8, 16, 32]},
        'lora_alpha': {'values': [16, 32, 64]},
        'target_modules' : {'values': [['v', 'q'], ['wi', 'v', 'wo', 'lm_head',
'k', 'o', 'q']]},
        'lora_dropout': {'values': [0.0, 0.05, 0.1]},
        'init_lora_weights': {'values': ["pissa", "olora"]},
    }
}
```

- QLoRA: T5 - base QLoRA Fietune Sweep

Python

```
qlora_sweep_configuration = {
    'method': 'random',
    'parameters': {
        'learning_rate': {'values': [5e-4, 1e-4, 2e-5]},
        'weight_decay' : {'values': [0.01, 0.0]},
        'r': {'values': [8, 16, 32]},
        'lora_alpha': {'values': [16, 32, 64]},
        'target_modules' : {'values': [['v', 'q'], ['wi', 'v', 'wo', 'lm_head',
'k', 'o', 'q']]},
        'lora_dropout': {'values': [0.0, 0.05, 0.1]},
    }
}
```

- IA3: T5 - base IA3 Fietune Sweep

Python

```
ia3_sweep_configuration = {
    'method': 'random',
    'parameters': {
        'learning_rate': {'values': [5e-4, 1e-4, 1e-5, 1e-3]},
        'weight_decay' : {'values': [0.01, 0.0]},
        'target_modules' : {'values': [['v', 'q'], ['wi', 'v', 'wo', 'lm_head',
'k', 'o', 'q']]},
```

```
    }
}
```

Evaluation

A avaliação sistemática de modelos de linguagem é fundamental em todas as etapas do desenvolvimento: antes, durante e após o treinamento. Em tarefas de Processamento de Linguagem Natural (NLP), onde é particularmente desafiador visualizar o comportamento dos dados e as respostas do modelo, precisamos ir além da análise qualitativa manual.

Embora a avaliação humana das predições seja valiosa, é necessário estabelecer métodos mais eficientes e objetivos para medir o desempenho. Isso garante uma avaliação consistente e escalável, permitindo comparações confiáveis entre diferentes versões do modelo e abordagens distintas.

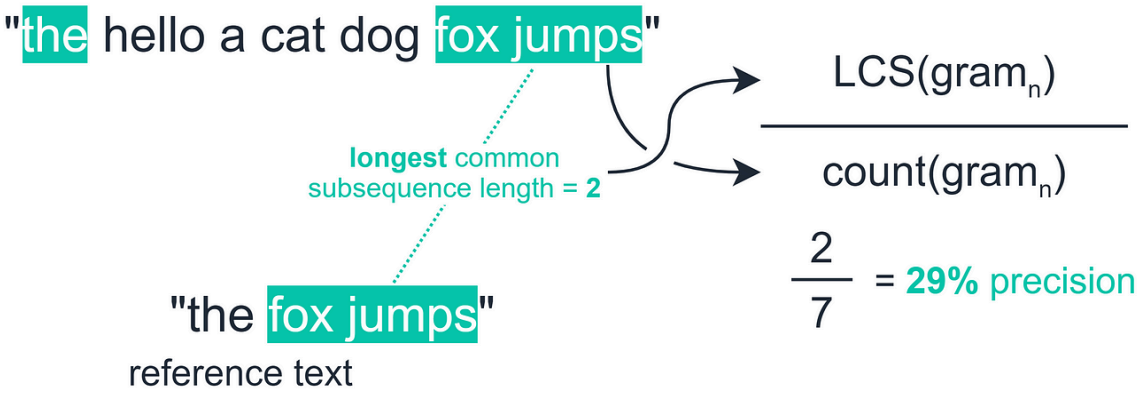
Métricas

ROUGE

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) é uma métrica que mede a sobreposição de palavras ou frases entre o texto gerado e uma referência. ROUGE examina vários conjuntos de palavras chamadas `n-grams`. As principais variantes incluem:

- ROUGE-1: Compara palavras individuais entre os textos, unigramas.
- ROUGE-2: Compara pares de palavras consecutivas, bigramas.
- ROUGE-L: Analisa a maior sequência de palavras que aparece em ambos os textos, capturando quão bem o modelo mantém a estrutura e a ordem dos elementos essenciais do texto.
- ROUGE-Lsum: Similar ao ROUGE-L, mas analisa cada frase separadamente antes de combinar os resultados.

Quanto maior o valor ROUGE (variando de 0 a 1), mais similar o resumo gerado é ao resumo de referência.



Fonte: [The Ultimate Performance Metric in NLP]

BERTScore

O [BERTScore] traz uma abordagem mais avançada para a avaliação de texto, utilizando embeddings gerados por modelos como o BERT para medir a similaridade semântica e contextual entre frases. Em vez de comparar apenas n-grams como as métricas tradicionais, o BERTScore analisa os `embeddings` de cada palavra nas sentenças, permitindo capturar nuances de significado e contexto que as outras métricas podem não perceber. Isso o torna especialmente útil para avaliar a qualidade de textos gerados por modelos que produzem linguagem fluida e com significado, mas sem uma correspondência exata de palavras.

G-Eval GPT-4

Utilizamos o [G-Eval], um método que avalia a qualidade do texto gerado por modelos de linguagem grandes (LLMs) sem precisar de uma referência padrão. Em vez disso, ele se baseia em critérios abstratos definidos em prompts de avaliação.

Os critérios são:

- **Relevância:** Assegura que o resumo contenha apenas informações essenciais, sem redundâncias.
- **Coerência:** Avalia a organização e o fluxo lógico do texto.
- **Consistência:** Verifica se o resumo é fiel aos fatos do documento original.
- **Fluência:** Analisa a gramática e a clareza do texto.

Criamos prompts específicos para cada critério, usando o texto original e o resumo como base, e orientamos o modelo a atribuir uma nota de 1 a 5. Usamos uma função direta onde o GPT-4 gera uma pontuação discreta para cada critério, garantindo uma avaliação abrangente e objetiva.

Resultados

Essa tabela demonstra os resultados obtidos ao realizar o finetuning do modelo t5-base utilizando as 4 técnicas escolhidas e selecionando o melhor modelo na busca de hiperparâmetros.

Métrica	t5-base	t5-base-full-finetune	t5-base-lora-finetune	t5-base-qlora-finetune	t5-base-ia3-finetune
rouge1	0.284943	0.402308	0.403291	0.418179	**0.424506**
rouge2	0.120233	0.177189	0.181117	0.196178	**0.211162**
rougeL	0.236728	0.332876	0.333803	0.348851	**0.359301**
rougeLsum	0.237333	0.330629	0.332017	0.348084	**0.356767**
precision	0.866779	0.871859	0.84034	**0.884034**	0.882415
recall	0.865186	**0.892520**	0.888285	0.889017	0.891723
f1	0.865903	0.883541	0.879876	0.886391	**0.886922**
relevance	4.280000	4.960000	5.280000	5.360000	**5.440000**
coherence	3.520000	3.720000	3.800000	4.360000	4.120000
consistency	5.880000	7.120000	7.080000	7.880000	7.040000
fluency	4.120000	4.240000	4.000000	4.640000	4.480000
all_params	222903552	222903552	**236923352**	153894144	223193728
trainable_params	**222903552**	**222903552**	14028800	1769472	290176
trainable_params_percent	**100.000000**	**100.000000**	5.921045	1.149798	0.130011
epoch	nan	3.000000	3.000000	3.000000	**5.000000**
train_runtime	nan	507.307100	574.073200	633.406300	**950.703200**
memory_footprint	0.830380	0.830380	**0.882642**	0.375546	0.831461

Além das métricas de avaliação, é possível observar a diferença na porcentagem de parâmetros utilizados em cada método, a quantidade de memória final dos adaptadores treinados e o tempo de treino em cada um.

Métrica	t5-base-full-finetune	t5-base-lora-finetune	t5-base-qlora-finetune	t5-base-ia3-finetune
rouge1	41.09% ↑	41.53% ↑	46.76% ↑	48.89% ↑
rouge2	47.37% ↑	50.64% ↑	63.16% ↑	75.63% ↑
rougeL	40.62% ↑	41.01% ↑	47.36% ↑	51.78% ↑
rougeLsum	39.31% ↑	39.89% ↑	46.66% ↑	50.32% ↑
precision	0.96% ↑	0.59% ↑	1.99% ↑	1.80% ↑
recall	3.16% ↑	2.67% ↑	2.75% ↑	3.07% ↑
f1	2.04% ↑	1.61% ↑	2.37% ↑	2.43% ↑
relevance	15.89% ↑	23.36% ↑	25.23% ↑	27.1% ↑
coherence	5.68% ↑	7.95% ↑	23.86% ↑	17.05% ↑
consistency	21.09% ↑	20.41% ↑	34.01% ↑	19.73% ↑
fluency	2.91% ↑	1.94%	12.62% ↑	8.74% ↑
all_params	0.0%	6.29% ↑	-30.96% ↓	0.13% ↑
trainable_params	0.0%	-93.71% ↓	-99.21% ↓	-99.87% ↓
trainable_params_percent	nan% ↓	-94.08% ↓	-98.85% ↓	-99.87% ↓
epoch	nan% ↓	nan% ↓	nan% ↓	nan% ↓
train_runtime	nan% ↓	nan% ↓	nan% ↓	nan% ↓
memory_footprint	0.0%	6.29% ↑	-54.77% ↓	0.13% ↑

Esta outra tabela demonstra a diferença em percentual entre os modelos treinados em relação ao modelo original pré-treinado.

Durante os experimentos, observamos que técnicas como LoRA e IA3 aumentaram o footprint de memória devido à adição de novos pesos adaptáveis à rede. Contudo, apenas uma pequena fração dos parâmetros foi ajustada (menos de 6%), permitindo que o modelo original permanecesse fixo enquanto adaptadores específicos eram aplicados para tarefas distintas. Essa estrutura modular facilita a inferência do modelo, permitindo a escolha de adaptadores conforme necessário para otimizar o desempenho em diferentes tarefas.

Avaliação Qualitativa

Para avaliar a eficácia do fine-tuning em modelos de linguagem, foi realizado um experimento que comparou as saídas de modelos ajustados em diferentes técnicas com o resumo humano de um diálogo de atendimento ao cliente. A análise mostrou como os modelos ajustados são capazes de entender e resumir diálogos complexos de forma mais precisa do que o modelo base, que apresentou dificuldades em capturar o contexto da conversa.

```
Python
--------------------------------------------------------------------------------
----------
PROMPT DE ENTRADA:
summarize: user: msg is not working on the app for the roku. My zip code is
14086
agent: I certainly apologize for any inconvenience Gregory. Has anyone been
able to look into this matter for you, or did you still need assistance?  .
user: Still need help This is my screen This is my screen
agent: Are you receiving this channel using your cable box?  .
user: No through the roku app
agent: Do you have a cable box in your home Gregory?  .
user: Yes I do, my wife is using it
agent: Can you please check to see if you are receiving MSG on the cable box?
.
user: I am
agent: Okay. Are you able to turn off the Roku &amp;amp; unplug your modem from the
electrical outlet? If so, please do so for approximately 20-30 seconds and go
ahead &amp;amp; re-plug the equipment.  .
user: Still not working
agent: If you'd like to follow us &amp;amp; DM your account details (phone number
and full address), I'd be happy to check into this for you.  .

--------------------------------------------------------------------------------
----------
SUMÁRIO HUMANO:
Customer is having an issue that the Ask_Spectrum message is not working on the
app for the roku. Agent updated the customer to check the message on the cable
box and informed to re-plug the equipment  and also inform to DM account
details further assist.
--------------------------------------------------------------------------------
----------
T5-BASE:
user: msg is not working on the app for the roku. user: my zip code is 14086.
--------------------------------------------------------------------------------
----------
T5-BASE-FULL-FINETUNE-TWEETSUMM:
```

```
Customer is complaining that msg is not working on the app for the roku. Agent
updated that to follow them and DM their account details (phone number and full
address).
--------------------------------------------------------------------------------
----------
T5-BASE-LORA-FINETUNE-TWEETSUMM:
Customer is complaining that msg is not working on the app for the roku. Agent
updated that if they would like to follow them &amp DM their account details
(phone number and full address), they would be happy to check into this.
--------------------------------------------------------------------------------
----------
T5-BASE-QLORA-FINETUNE-TWEETSUMM:
Customer is complaining that msg is not working on the app for the roku. Agent
updated that if they would like to follow us &amp; DM their account details
(phone number and full address), they would be happy to check into this for
them.
--------------------------------------------------------------------------------
----------
T5-BASE-IA3-FINETUNE-TWEETSUMM:
Customer is complaining that msg is not working on the app for the roku. Agent
updated that if they would like to follow them and DM their account details
(phone number and full address), he would be happy to check into this for them.
--------------------------------------------------------------------------------
----------
```

Os resumos dos modelos ajustados apresentam um formato mais próximo ao resumo humano, com uma estrutura de linguagem padronizada e informativa. Neste exemplo, as diferenças principais estão nos conectivos e na fluidez da resposta. Mesmo com um dataset de treinamento limitado, as técnicas de ajuste fino demonstraram um bom desempenho ao capturar a essência do diálogo.

As experimentações confirmaram a eficácia das técnicas de Parameter-Efficient Fine-Tuning (PEFT), para ajustar modelos de linguagem com eficiência, mesmo em cenários com modelos menores e datasets limitados. Embora os ganhos sejam modestos em modelos pequenos, o impacto dessas técnicas tende a aumentar significativamente à medida que o tamanho do modelo cresce, tornando-as altamente promissoras para arquiteturas maiores.

Técnicas como LoRA e QLoRA revelam um grande potencial de escalabilidade. Elas oferecem melhorias contínuas à medida que o tamanho do modelo aumenta, permitindo que uma fração dos parâmetros seja ajustada sem comprometer a qualidade do ajuste fino. Essa abordagem mantém a eficiência computacional, que é essencial em cenários com recursos limitados.

A eficácia dessas técnicas depende de fatores como o tamanho do modelo, a estratégia de quantização, o volume de dados e a capacidade da GPU disponível.

Conclusão

O fine-tuning é uma técnica poderosa, mas deve ser aplicada com cautela. Embora possa oferecer grandes benefícios, sua implementação exige recursos computacionais significativos, além de tempo, estudo e capacitação técnica. É importante avaliar cada cenário para determinar se o fine-tuning é realmente necessário ou se alternativas mais simples poderiam alcançar a otimização desejada com menor esforço.

As técnicas PEFT representam uma alternativa eficiente ao full fine-tuning, especialmente no que diz respeito aos recursos computacionais. Compreender as características de cada método e o comportamento dos parâmetros envolvidos é essencial para alcançar bons resultados e tomar decisões informadas, minimizando desperdícios de tempo e dinheiro.

A combinação das ferramentas peft, Transformers, Hugging Face Hub e Weights & Biases configura uma abordagem robusta e eficiente para o ajuste fino de modelos. Essa estratégia abrangente não apenas facilita o processo, mas também oferece uma estrutura sólida para alcançar um ajuste de alta qualidade e manter controle sobre o desempenho dos modelos em diferentes configurações.