UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

TOBIAS GONÇALVES PIRES

Análise de técnicas de ajuste fino em classificação de texto

GOIÂNIA
2024

UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO (TECA) PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TESES

E DISSERTAÇÕES NA BIBLIOTECA DIGITAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio da Biblioteca Digital de Teses e Dissertações (BDTD/UFG), regulamentada pela Resolução CEPEC nº 832/2007, sem ressarcimento dos direitos autorais, de acordo com a Lei 9.610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou download, a título de divulgação da produção científica brasileira, a partir desta data.

O conteúdo das Teses e Dissertações disponibilizado na BDTD/UFG é de responsabilidade exclusiva do autor. Ao encaminhar o produto final, o autor(a) e o(a) orientador(a) firmam o compromisso de que o trabalho não contém nenhuma violação de quaisquer direitos autorais ou outro direito de terceiros.

1. Identificação do material bibliográfico

[x] Dissertação [ ] Tese [ ] Outro*:_____

*No caso de mestrado/doutorado profissional, indique o formato do Trabalho de Conclusão de Curso, permitido no documento de área, correspondente ao programa de pós-graduação, orientado pela legislação vigente da CAPES.

Exemplos: Estudo de caso ou Revisão sistemática ou outros formatos.

2. Nome completo do autor

Tobias Gonçalves Pires

3. Título do trabalho

Análise de técnicas de ajuste fino em classificação de texto

4. Informações de acesso ao documento (este campo deve ser preenchido pelo orientador)

Concorda com a liberação total do documento [X] SIM [ ] NÃO[1]

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. Após esse período, a possível disponibilização ocorrerá apenas mediante:
a) consulta ao(à) autor(a) e ao(à) orientador(a);
b) novo Termo de Ciência e de Autorização (TECA) assinado e inserido no arquivo da tese ou dissertação.
O documento não será disponibilizado durante o período de embargo.
Casos de embargo:
- Solicitação de registro de patente;
- Submissão de artigo em revista científica;
- Publicação como capítulo de livro;
- Publicação da dissertação/tese em livro.

Obs. Este termo deverá ser assinado no SEI pelo orientador e pelo autor.

sei! assinatura eletrônica

Documento assinado eletronicamente por **Anderson Da Silva Soares**, **Professor do Magistério Superior**, em 08/10/2024, às 16:53, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

Documento assinado eletronicamente por **Tobias Gonçalves Pires**, **Discente**, em 08/10/2024, às 17:03, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **4887471** e o código CRC **854DFEB6**.

TOBIAS GONÇALVES PIRES

Análise de técnicas de ajuste fino em classificação de texto

Dissertação apresentada ao Programa de Pós–Graduação em ciência da computação do Instituto de Informática da Universidade Federal de Goiás, como requisito para obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Ciência da Computação.

Linha de Pesquisa: Sistemas Inteligentes e Aplicações.

Orientador: Dr. Anderson da Silva Soares

GOIÂNIA
2024

Ficha de identificação da obra elaborada pelo autor, através do
Programa de Geração Automática do Sistema de Bibliotecas da UFG.

Pires, Tobias Gonçalves
Análise de técnicas de ajuste fino em classificação de texto [manuscrito] / Tobias Gonçalves Pires. - 2024.
77 f.

Orientador: Prof. Dr. Anderson da Silva Soares.
Dissertação (Mestrado) - Universidade Federal de Goiás, Instituto de Informática (INF), Programa de Pós-Graduação em Ciência da Computação, Goiânia, 2024.

Inclui siglas, lista de figuras, lista de tabelas.

1. . I. Soares, Anderson da Silva, orient. II. Título.

CDU 004

UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

ATA DE DEFESA DE DISSERTAÇÃO

Ata nº **26** da sessão de Defesa de Dissertação de **Tobias Gonçalves Pires**, que confere o título de Mestre em Ciência da Computação, na área de concentração em Ciência da Computação.

Aos vinte e três dias do mês de setembro de dois mil e vinte e quatro, a partir das catorze horas, via webconferência, realizou-se a sessão pública de Defesa de Dissertação intitulada **"Análise de técnicas de ajuste fino em classificação de texto"**. Os trabalhos foram instalados pelo Orientador, Professor Doutor Anderson da Silva Soares (INF/UFG) com a participação dos demais membros da Banca Examinadora: Professor Doutor Rodrigo Zempulski Fanucchi (COPEL Distribuição S.A.), membro titular externo; Professor Doutor Arlindo Rodrigues Galvão Filho (INF/UFG), membro titular interno. A realização da banca ocorreu por meio de videoconferência. Durante a arguição os membros da banca não fizeram sugestão de alteração do título do trabalho. A Banca Examinadora reuniu-se em sessão secreta a fim de concluir o julgamento da Dissertação, tendo sido o candidato **aprovado** pelos seus membros. Proclamados os resultados pelo Professor Doutor Anderson da Silva Soares, Presidente da Banca Examinadora, foram encerrados os trabalhos e, para constar, lavrou-se a presente ata que é assinada pelos Membros da Banca Examinadora, aos vinte e três dias do mês de setembro de dois mil e vinte e quatro.
TÍTULO SUGERIDO PELA BANCA

Documento assinado eletronicamente por **Anderson Da Silva Soares**, **Professor do Magistério Superior**, em 23/09/2024, às 16:17, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Arlindo Rodrigues Galvao Filho**, **Professor do Magistério Superior**, em 23/09/2024, às 16:45, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **RODRIGO ZEMPULSKI FANUCCHI**, **Usuário Externo**, em 23/09/2024, às 18:04, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Tobias Gonçalves Pires**, **Discente**, em 25/09/2024, às 08:49, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **4761500** e o código CRC **5540DC66**.

Referência: Processo nº 23070.039764/2024-42 SEI nº 4761500

Todos os direitos reservados. É proibida a reprodução total ou parcial do trabalho sem autorização da universidade, do autor e do orientador(a).

Tobias Gonçalves Pires

Graduou–se em Sistemas de Informação na ALFA - Faculdade Alves Faria. Experiência profissional com desenvolvimento, projeto e arquitetura de software, atuou em diversas empresas privadas inclusive multinacional como Indra Politec. Atualmente trabalha desenvolvimento soluções de Big Data e Processamento de Linguagem Natural.

Dedico esta dissertação aos meus pais, que sempre me proporcionaram uma educação exemplar e demonstraram através de suas próprias jornadas acadêmicas e profissionais, a importância dos estudos e do trabalho

Agradecimentos

A realização desta dissertação de mestrado foi um desafio recompensador e um marco significativo em minha trajetória acadêmica e profissional. Muitas pessoas contribuíram de diversas maneiras para que eu pudesse alcançar este objetivo, e gostaria de expressar minha gratidão.

Primeiramente, agradeço ao meu pai, cujo incentivo foi fundamental para que eu iniciasse este mestrado. Sua disposição em ajudar na revisão do texto da dissertação foi inestimável, contribuindo com a clareza e a qualidade deste trabalho.

Agradeço ao meu orientador, Professor Anderson, pela oportunidade de participar de projetos na área de inteligência artificial, que é o foco desta dissertação. Sua orientação, expertise e apoio foram essenciais para o desenvolvimento deste trabalho.

Resumo

Pires, Tobias. **Análise de técnicas de ajuste fino em classificação de texto**. GOIÂNIA, 2024. 79p. Dissertação de Mestrado. Instituto de Informática, Universidade Federal de Goiás.

O Processamento de Linguagem Natural (PLN) visa desenvolver modelos que capacitem computadores a compreender, interpretar, processar e gerar texto de maneira semelhante à comunicação humana. A última década presenciou avanços significativos na área com a introdução de modelos de redes neurais profundas. A subsequente evolução da arquitetura desses modelos como o mecanismo de atenção e a arquitetura Transformers, culminou em modelos de linguagem como ELMo, BERT e GPT. E, posteriormente modelos denominados grandes modelos de linguagem que melhoraram a capacidade de entender e gerar textos de forma sofisticada. Modelos pré-treinados oferecem a vantagem de reutilizar conhecimento acumulado de vastos conjuntos de dados, embora ajustes finos específicos sejam necessários para tarefas individuais. No entanto, o treinamento e ajuste desses modelos consomem muitos recursos de processamento, tornando-se inviáveis para muitas organizações devido aos altos custos. Em ambientes com restrições de recursos, técnicas de ajuste fino eficientes como o LoRA (Low-Rank Adaptation) foram desenvolvidas para otimizar o processo de adaptação de modelos, minimizando a quantidade de parâmetros ajustáveis e no caso do LoRA evitando o *overfitting* [19]. Essas técnicas permitem um treinamento mais rápido e econômico, mantendo a robustez e generalização dos modelos. Este trabalho avalia três técnicas de ajuste fino eficientes LoRA, AdaLoRA e IA3 (além do ajuste fino completo) em termos de consumo de memória, tempo de treinamento e acurácia, utilizando os modelos DistilBERT, Roberta-base e TinyLlama em três diferentes datasets AG News (AG's News Corpus), IMDb e SNLI (Stanford Natural Language Inference).

Palavras–chave

Processamento de Linguagem Natural, Bidirectional Encoder Representations for Transformers, Embeddings from Language Models, Generative Pre-trained transformer, Low-Rank Adaptation, Adaptative Low-Rank Adaptation, Internet Movie Database, Stanford Natural Language Inference

Abstract

Pires, Tobias. **Analysis of fine-tuning techniques in text classification**. GOIÂNIA, 2024. 79p. MSc. Dissertation. Instituto de Informática, Universidade Federal de Goiás.

Natural Language Processing (NLP) aims to develop models that enable computers to understand, interpret, process and generate text in a way similar to human communication. The last decade has seen significant advances in the field, with the introduction of deep neural network models, and the subsequent evolution of the architecture of these models such as the attention mechanism and the Transformers architecture, culminating in language models such as ELMo, BERT and GPT. And later models called Large Language Models (LLMs) improved the ability to understand and generate texts in a sophisticated way. Pre-trained models offer the advantage of reusing knowledge accumulated from vast datasets, although specific fine-tuning is required for individual tasks. However, training and tuning these models consumes a lot of processing resources, making it unfeasible for many organizations due to high costs. In resource-constrained environments, efficient fine-tuning techniques such as LoRA (Low-Rank Adaptation) were developed to optimize the model adaptation process, minimizing the number of adjustable parameters and avoiding overfitting. These techniques allow for faster and more economical training, while maintaining the robustness and generalization of the models. This work evaluates three efficient fine-tuning techniques LoRA, AdaLoRA and IA3 (in addition to full fine-tuning) in terms of memory consumption, training time and accuracy, using the DistilBERT, Roberta-base and TinyLlama models on different datasets (AG News, IMDb and SNLI).

Keywords

Natural Language Processing, Large Language Models, Bidirectional Encoder Representations for Transformers, Embeddings from Language Models, Generative Pre-trained transformer, Low-Rank Adaptation, Adaptative Low-Rank Adaptation, Internet Movie Database, Stanford Natural Language Inference

Sumário

Lista de Figuras **14**

Lista de Tabelas **15**

1 Introdução **16**
- 1.1 Objetivos geral e específicos 19
- 1.2 Justificativa 19
- 1.3 Metodologia 20
- 1.4 Estrutura do Documento 20

2 Fundamentação teórica e trabalhos correlatos **21**
- 2.1 Trabalhos Correlatos 21
- 2.2 Rede Neural Recorrente 22
- 2.3 Processamento de Linguagem Natural 23
- 2.4 Modelo de Linguagem 25
- 2.5 Ajuste fino 27
- 2.6 Arquitetura Transformers 28
  - 2.6.1 Camada de Atenção 30
  - 2.6.2 Conexões Residuais 30
  - 2.6.3 Camada de Normalização 30
  - 2.6.4 Camada *Feedforward* 31
  - 2.6.5 Codificador e Decodificador 31
- 2.7 BERT 31
  - 2.7.1 Pré-Processamento de Dados 31
    - Preenchimento de Lacunas (*Masked Language Model - MLM*) 32
    - Previsão da Próxima Sentença - *Next Sentence Prediction (NSP)* 32
  - 2.7.2 Estrutura de Entrada 33
  - 2.7.3 Codificação Bidirecional 33
  - 2.7.4 Ajuste fino do BERT para classificação de texto 34
- 2.8 DistilBERT 35
  - 2.8.1 *Distillation loss* 36
  - 2.8.2 *Classic loss* 36
  - 2.8.3 DistilBERT vs BERT 36
- 2.9 RoBERTA 37
- 2.10 Grandes Modelos de Linguagem 37
- 2.11 Llama 39
- 2.12 Tiny Llama 39
- 2.13 Técnicas para redução de modelo 40

2.13.1 Prunning 40
2.13.2 Quantization 40
2.14 Técnicas de Ajuste fino com eficiência de parâmetros 41
2.14.1 Métodos aditivos 41
Adapter-like 41
Soft prompts 41
2.14.2 Métodos seletivos 41
2.14.3 Métodos baseados em reparametrização 42
2.14.4 Métodos Híbridos 42
2.15 Conjuntos de dados 43
2.15.1 IMDb 43
2.15.2 AG News 43
2.15.3 SNLI 44
2.16 Transferência de aprendizado 45
2.17 LoRA (*Low-Rank Adaptation*) 46
2.17.1 Apresentação do Problema 46
2.17.2 Funcionamento do LoRA na atualização dos pesos 47
Experimentos realizados 49
2.18 AdaLora (*Adaptive Low-Rank Adaptation*) 50
2.18.1 Experimentos 52
DeBERTaV3-base + GLUE 52
BART-large + XSum e CNN/DailyMail 53
2.19 IA3 (*Infused Adapter by Inhibiting and Amplifying Inner Activations*) 54
2.19.1 Detalhes do IA3 55
Treinamento de improbabilidade (*Unlikelihood Training*) 56
Normalização de comprimento (*Length Normalization*) 56
Resultados no conjuntos de dados RAFT (Real-world Few-shot Tasks) 56

3 A Abordagem proposta neste trabalho para os Experimentos 58
3.1 Técnicas e recursos a serem utilizados nos experimentos 58
3.2 Aspectos a serem avaliados nos experimentos 59
3.3 Medidas a serem obtidas dos experimentos 59
3.4 Recursos computacionais para realização dos experimentos 60

4 Resultados e Análises 62
4.1 Resultados e Análises do uso de memória da GPU 62
4.2 Resultados e Análises do tempo de processamento dos experimentos 64
4.3 Resultados e Análises da acurácia obtida nos experimentos 65
4.4 Resultados e Análises do processo de aprendizado ocorrido nos experimentos 66
4.5 Resultados e Análises do quantitativo de parâmetros treináveis 68
4.6 Contribuições do trabalho 68
4.6.1 Principais Contribuições 69
Avaliação de Modelos com Diferentes Capacidades 69
Eficácia das Técnicas de Ajuste Fino 69
Comparação de Desempenho em Diferentes Conjuntos de Dados 69
Análise de Consumo de Memória e Tempo de Treinamento 69
Impacto na Acurácia 69
Considerações Finais 69

5 Conclusões e Trabalhos Futuros **71**
5.1 Conclusões 71
5.2 Trabalhos Futuros 72

Referências Bibliográficas **74**

Lista de Figuras

1.1 Data de lançamento e tamanho de alguns dos modelos, fonte: [56]. 17

2.1 Fluxo básico da informação em uma RNN. 22
2.2 Estrutura básica de uma célula LSTM. 23
2.3 Visão Geral da arquitetura *Transformer*. 29
2.4 Relação entre tokens em duas cabeças de atenção. 30
2.5 Fluxo de aplicação de máscara e previsão do token no BERT. 32
2.6 Fluxo de previsão de próxima sentença no BERT. 33
2.7 Visão geral da estrutura de entrada de dados no BERT. 33
2.8 Processo de classificação no ambiente do BERT. 34
2.9 Ciclo do processo de destilação. 35
2.10 Técnicas existentes de PEFT em seus respectivos grupos. **fonte:** Imagem retirada de [29]. 42
2.11 Esquema de atualização das matrizes com o LoRA 48
2.12 Acurácia do LoRA em diferentes configurações, fonte: [66]. 51
2.13 Estrutura básica do IA3 fonte:[30] 55

Lista de Tabelas

2.1 Trecho do conjunto de dados IMDb. 43
2.2 Trecho do conjunto de dados AG News. 44
2.3 Trecho do conjunto de dados SNLI. 45
2.4 Resultados dos experimentos com diferentes combinações de matrizes e r 49
2.5 Resultados dos experimentos com diferentes combinações de matrizes e r 50
2.6 Testes com o modelo DeBERTaV3-base + conjunto de dados GLUE 52
2.7 Testes com o modelo DeBERTaV3-base + conjunto de dados SQuAD 53
2.8 Testes com o modelo BART-large + conjuntos de dados XSum e CNN/DailyMail 54
2.9 Comparação da técnica IA3 com vários outras técnicas em vários datasets. 57

4.1 Resultados de uso de memória em gigabyte por experimento. 63
4.2 Resultados de tempo dispendido, em minutos, para o processamento dos experimento. 64
4.3 Resultados das acurácias, em %, obtidas nos experimentos. 65
4.4 Resultados das acurácias obtidas nos dados de treino e teste por épocas no modelo DistilBERT. 66
4.5 Resultados das acurácias obtidas nos dados de treino e teste por épocas no modelo RoBERTa. 67
4.6 Resultados das acurácias obtidas nos dados de treino e teste por épocas no modelo TinyLlama. 67
4.7 Quantidade de parâmetros passíveis de ajuste em cada técnica de ajuste fino 68

CAPÍTULO 1

Introdução

A área de Processamento de Linguagem Natural (PLN) tem como objetivo principal desenvolver modelos e sistemas que permitam aos computadores compreender, interpretar, processar e gerar texto [22]. O PLN busca analisar a linguagem de forma profunda para entender o significado, contexto e intenção por trás das expressões linguísticas, permitindo que os computadores manipulem a linguagem humana de maneira mais eficiente e similar à comunicação humana. Na última década vimos um desenvolvimento expressivo das técnicas de PLN [22].

Em 2013, os trabalhos de Tomas Mikolov [35], [36] propuseram um modelo de Redes Neurais Artificiais (RNA) para geração de uma representação multidimensional para as palavras de uma língua, estabelecendo uma proximidade espacial de palavras com semânticas parecidas. Em 2014 Mikolov buscou a mesma representação com sentenças e documentos [27], encontrando dificuldades em seus experimentos como perda de informação e ambiguidade semântica. Seguiu-se outros trabalhos importantes como a proposta do mecanismo de *attention* [58], o qual resultou na arquitetura *Transformer* [58]. Esses trabalhos, entre outros, levaram a avanços significativos no campo de PLN. Apareceram os primeiros modelos de linguagem como o *Embeddings from Language Model* (ELMo) [41], *Bidirectional Encoder Representations for Transformers* (BERT) [12], *Generative Pre-trained Transformer 1* (GPT-1) [43]e na sequência os modelos de grande escala referidos como *Large Language Models* (LLMs). Esses modelos representam uma evolução importante na capacidade de entender e gerar textos de maneira mais sofisticada, aproximando-se cada vez mais da complexidade da linguagem humana.

Existem algumas vantagens em se utilizar um modelo pré-treinado (ou modelo fundacional) como base, pois estes foram treinados com quantidade massivas de dados e possuem um conhecimento da linguagem que pode ser reaproveitado em outras várias tarefas, embora seja necessário fazer o ajuste fino do modelo para cada tarefa específica [18]. Esse procedimento será discutido e detalhado mais a diante neste trabalho.

Atualmente diversos modelos de linguagem estão disponíveis para solução de inúmeros tipos de problemas em PLN. Entretanto, treinar os grandes modelos de linguagem consome muito recurso de processamento [14], [63]. O tamanho dos modelos de

linguagem vem aumentando de forma rápida, e o custo do hardware para se beneficiar das soluções estado da arte para a maior parte das empresas tem um custo proibitivo, sendo acessível para um grupo limitado de empresas, e ainda sim, a cada ano que se passa os modelos lançados estão cada vez maiores e o custo se torna cada vez mais inacessível. Abaixo gráfico com a data de lançamento e tamanho de alguns dos modelos lançados nos últimos anos.

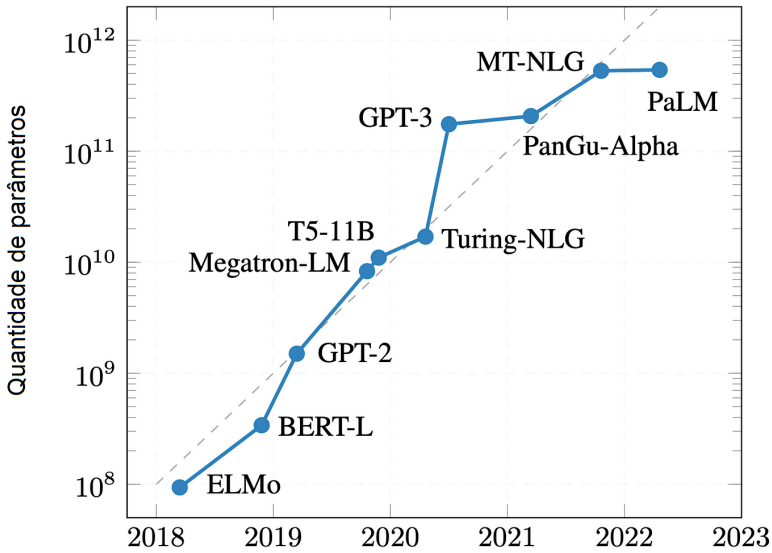

Figura 1.1: Data de lançamento e tamanho de alguns dos modelos, fonte: [56].

Isso ocorre devido ao crescimento da necessidade de uso de hardware em *deep learning* que demanda de forma direta o consumo de mais energia elétrica. Esse consumo tem impacto direto nas emissões de carbono, levantando, inclusive, questionamentos relacionados a sustentabilidade desses modelos [32] [39]. Portanto, este trabalho se ocupa com os recursos de máquina que, naturalmente, terão consequências na eficiência energética.

Em um ambiente com limitações de recursos (financeiros e de infraestrutura computacional) para realização de pesquisas, se torna necessário a realização de estudos e avanços de técnicas que possam tornar todo o processo de aprendizado de máquina mais rápido, barato e eficiente. Nesse sentido, as técnicas de ajuste fino com eficiência na manipulação de parâmetros foram desenvolvidas e visam otimizar o processo de adaptação do modelo, permitindo que ele seja ajustado de forma mais rápida para novos conjuntos de dados. Ao minimizar a quantidade de parâmetros a serem ajustados durante o treinamento, as técnicas de Ajuste fino com eficiência de parâmetros, como o *Low-Rank Adaptation* (LoRA) [19], garantem que armazenamento dos módulos ajustados sejam compactos e mais adequados para implantação em ambientes com recursos restritos.

O uso de técnicas de ajuste fino com eficiência de parâmetros possibilita evitar o *overfitting* [29], um problema comum em aprendizado profundo, onde o modelo se

adapta demasiadamente aos dados de treinamento, perdendo a capacidade de generalizar seu comportamento para novos exemplos. Ao restringir a quantidade de parâmetros a serem ajustados e incorporar regularização durante o treinamento, essas técnicas também ajudam a mitigar esse problema, resultando em modelos mais robustos e generalizáveis. Além disso, o uso de técnicas de ajuste fino com eficiência de parâmetros em aprendizado profundo é justificado pelas vantagens que oferecem, tais como: aproveitamento de conhecimento prévio, eficiência computacional e adaptação mais rápida a uma nova tarefa. Enfim, essas técnicas desempenham um papel fundamental na construção de soluções de aprendizado profundo, eficazes e de alto desempenho para uma variedade de aplicações práticas.

Uma técnica de ajuste fino que vem propiciando diversos trabalhos de aprimoramento da técnica desde sua proposta em 2021 é o LoRA [19], essa técnica acelera o ajuste fino de modelos de linguagem pré-treinados. Descrevendo de forma bem resumida, o LoRA adiciona matrizes de parâmetros relativamente pequenas no modelo, treinando somente os parâmetros dessas matrizes e deixando congelado os parâmetros do modelo original. Com essa abordagem, o LoRa reduz vários dos problemas existentes em técnicas correlatas, algumas das quais são detalhadas adiante neste trabalho. O LoRA tenta fazer o ajuste fino de um modelo ter um desempenho satisfatório treinando um conjunto menor de parâmetros, o que pode reduzir significativamente o custo do projeto e permitir, por exemplo, que pesquisadores que trabalham com outras línguas que não o inglês, consigam atingir resultados mais próximos do estado da arte do inglês.

Após a contextualização do ambiente onde este trabalho se situa, pode-se delimitar o foco de atuação dele. Este trabalho avaliou três técnicas de ajuste fino, sendo elas: Low-Rank Adaptation (LoRA) [19], *Adaptive Budget Allocation* (AdaLoRA) [66] e Infused Adapter by Inhibiting and Amplifying Inner Activations (IA3) [30]. As duas primeiras técnicas seguem uma abordagem semelhante, enquanto a terceira tem uma abordagem diferente das outras duas. O objetivo foi avaliar a eficiência de uso de parâmetros, o consumo de memória, o tempo de treinamento e a acurácia de cada uma dessas três técnicas. Foram utilizados três modelos de linguagem para os testes, sendo eles o *distilled version of BERT* (DistilBERT) [48], *Robustly Optimised BERT Approach* (Roberta-base) [31] e o TinyLlama [65], sendo este último significativamente maior que os outros dois. Para os testes serão utilizados os datasets *AG's News Corpus* (AG News) [67], *Internet Movie Database* (IMDb) [11] e *Stanford Natural Language Inference* (SNLI) [5]. Também, realizou-se um ajuste fino completo a fim de permitir uma análise comparativa mais detalhada dos experimentos realizados.

1.1 Objetivos geral e específicos

Na contextualização apresentada na introdução deste trabalho, pode-se concluir que o objetivo geral dele é realizar experimentos de aplicação de técnicas de ajuste fino com eficiência de parâmetros em modelos de linguagem de diferentes tamanhos, permitindo a análise dos resultados dos experimentos e o registros científico dessas análises.

Os objetivos específicos para o projeto são:

- Estudar e avaliar as técnicas de ajuste fino com eficiência de parâmetros LoRA, AdaLoRA e IA3.
- Aplicar essas técnicas em modelos de linguagem com diferentes tamanhos, sendo eles: DistilBERT (66M), Roberta-base (125M) e TinyLlama (1Bi).
- Implementar experimentos de ajuste fino com os datasets: AG News, IMDb e SNLI.
- Implementar experimentos de ajuste fino completo dos modelos de linguagem com os mesmos datasets utilizados nas outras técnicas de ajuste fino.
- Comparar os resultados dos trinta e seis experimentos realizados, utilizando critérios: tempo de treinamento, uso de memória e acurácia.
- Estabelecer análises com os resultados que possam indicar as melhores abordagens para determinadas aplicações ou cenários.
- Registrar em texto científico (dissertação), todos os detalhes necessários para o entendimento e reprodução do resultados obtidos neste trabalho.

1.2 Justificativa

Avaliando as atividades proposta para este trabalho, pode-se observar que serão obtidos resultados de trinta e seis experimentos envolvendo o uso de técnicas de ajuste fino. Tais resultados envolverão algumas técnicas recentes de ajuste fino e modelos de linguagem com tamanhos bastante diferentes, isto é, um conjunto de combinações com diferentes vantagens e desvantagens a depender do cenário e necessidades.

Assim, este trabalho se justifica por diversos aspectos, entre eles, os principais seriam:

1. A publicação de resultados que possam orientar outros pesquisadores e ou profissionais de mercado na escolha e utilização das técnicas de ajuste fino em suas aplicações específicas.
2. Demonstrar a efetividade de diferentes técnicas de ajuste fino em modelos de diferentes tamanhos, visto que, como foi demostrado em outros trabalhos [29], diferentes técnicas podem ter resultados significativamente diferentes a depender do tamanho do modelo.

3. A formação de pesquisador habilitado na área de atuação em PLN, podendo este multiplicar o aprendizado obtido no trabalho.
4. Produção de artigos científicos com resultados das comparações estimulando o aprimoramento de técnicas de ajuste fino.

1.3 Metodologia

O processo metodológico deste trabalho se resume, basicamente, em planejamento e execução de cada um dos passos descritos nos objetivos específicos. Com a finalidade de atender o objetivo geral, os objetivos específicos foram ordenados em ordem temporal de modo que permitiu adquirir conhecimentos fundamentais acerca dos assuntos que envolvem este trabalho. Dessa forma, partiu-se do estudos sobre a arquitetura *Transformer*, passando pelo entendimento das técnicas de ajuste fino, e a habilitação tecnológica para a implementação dos experimentos.

A escolha dos modelos de linguagem e dos conjuntos de dados para treinamento foram feitos por meio de exploração bibliográfica, leitura e seleção de trabalhos que tratam dos assuntos, seguidos da escolha e justificativa destas. Após essa etapa, os experimentos foram implementados, executados e os resultados coletados para análises. Os detalhes da implementação, tabelas de resultados e o processo de análise dos resultados estão detalhados mais adiante deste trabalho.

1.4 Estrutura do Documento

Além deste primeiro Capítulo contendo a introdução, objetivos e justificativas, esta dissertação apresenta no Capítulo 2 um referencial teórico, detalhando os conhecimentos necessários adquiridos para a realização do trabalho, bem como alguns trabalhos correlatos a este. No Capítulo 3, os experimentos são organizados de modo que o processo metodológico é detalhado a fim de justificar a abordagem adotada nos experimentos, bem como os critérios de análise dos resultados. No capítulo 4, são apresentados os resultados em tabelas e as análises realizadas dos experimentos realizados. Finalmente, o Capítulo 5 apresenta as conclusões obtidas no trabalho e enumera algumas sugestões de trabalhos futuros.

CAPÍTULO 2

Fundamentação teórica e trabalhos correlatos

Este capítulo apresenta alguns conceitos básicos necessários ao entendimento e à realização dos experimentos, análises e discussões apresentadas nos capítulos 3 e 4 deste trabalho. Antes porém, uma vez entendido o contexto e o objetivo, será apresentado alguns trabalhos correlatos a fim de reafirmar a importância e originalidades desta dissertação.

2.1 Trabalhos Correlatos

Esta seção apresenta alguns trabalhos cujo tema está próximo a este. Existem inúmeros trabalhos comparativos de performance de redes neurais na literatura. Assim, esta seção buscou apresentar trabalhos realmente correlatos que efetuam testes de eficiência em LLM's.

No trabalho de Zhang e outros, em 2022 [66], foram realizados uma série de testes para validar a eficácia do AdaLoRA, comparando esta técnica com outras como o *LoRA, BitFit, Houlsby Adapter* e *Pfeiffer Adapter*. Zhang e outros também fizeram ajuste fino completo nos modelos *DeBERTaV3-base* e *BART-large*. E também, utilizou um conjunto de dados GLUE que avalia a compreensão da linguagem natural, dois conjuntos para perguntas e respostas (SQuADv1 e SQuADv2) e dois conjuntos para geração de linguagem natural (XSum e CNN/DailyMail). E, finamente, para ter mais base de comparação foram feitos testes com diferentes quantidade de parâmetros treináveis para cada uma das técnicas de ajuste fino, como 0.33M aproximadamente e 1.33M.

O trabalho [17] propõe uma abordagem para o aprendizado por transferência em PLN, utilizando módulos adaptadores (Adapters) para tornar o processo mais eficiente em termos de parâmetros. Testes foram realizados em 26 tarefas de classificação de texto, incluindo o benchmark GLUE. No GLUE, os adaptadores atingiram um desempenho próximo ao estado da arte, ficando apenas 0.4% abaixo do fine-tuning completo, mas com muito menos parâmetros treináveis (3% a 8% do total).

O trabalho de Houlsby e outros, em 2019 [13], faz uma comparação de algumas técnicas de ajuste fino, sendo estas o prompt tuning, o prefix-tuning, o LoRA, o Adapters e ajuste fino completo. Os testes foram feitos em aproximadamente 100 tarefas de PLN

e no modelo T5-Base. O estudo concluiu que em muitos casos isolados, as técnicas de ajuste fino eficientes são bem piores que o ajuste fino completo. Entretanto, fazendo uma média geral, a diferença não foi tão grande.

Em alguns poucos casos, as técnicas de ajuste fino eficientes foram superiores ao ajuste fino completo, o que demonstra um importante potencial de aplicação das técnicas de ajuste fino eficientes a depender do cenário. Em termos de resultados médios as técnicas de ajuste fino ficaram na seguinte ordem da melhor para a pior: ajuste fino completo, LoRA, Adapters, Prompt tuning e Prefix-tuning.

O trabalho de Wu e outros, em 2024 [62], compara as técnicas de ajuste fino Prefix Tunning, Series Adapters, Parallel Adapters, LoRA, DoRA e LoReFT, com os modelos LLaMA-7B e LLaMA-13B. No trabalho utilizou-se os datasets BoolQ, PIQA, SIQA, HellaS., WinoG., ARC-e, ARC-c, OBQA, AQuA, GSM8K, MAWPS e SVAMP.

O LoReFT desenvolvido no trabalho de Wu e outros [62], apresentou os melhores resultados nos testes. Tendo uma acurácia de 83.3% contra 81.5% do DoRa e 80.5% do LoRA no LLaMA-13B. E, no caso do LLaMA-7B a vantagem do LoReFT fica ainda maior, tendo uma acurácia de 80.2% contra 78.1% do DoRa e 74.7% do LoRA.

2.2 Rede Neural Recorrente

Uma rede recorrente é um tipo de rede neural especialmente adequada para tarefas que envolvem dados sequenciais ou temporais. Essas redes possuem conexões que formam ciclos, permitindo que a informação persista e seja reutilizada ao longo das iterações [16], diferente das redes neurais *feedforward* convencionais, onde os dados fluem apenas em uma direção (da entrada para a saída). Isso torna as RNNs particularmente poderosas para tarefas onde o contexto ou dados anteriores são importantes, embora estas tenham problemas relacionados a gradientes desaparecendo e explodindo, e com dependências de longo prazo. A figura 2.1 mostra o fluxo de dados em uma rede recorrente onde parte das informações iniciais são enviadas para as informações seguintes.

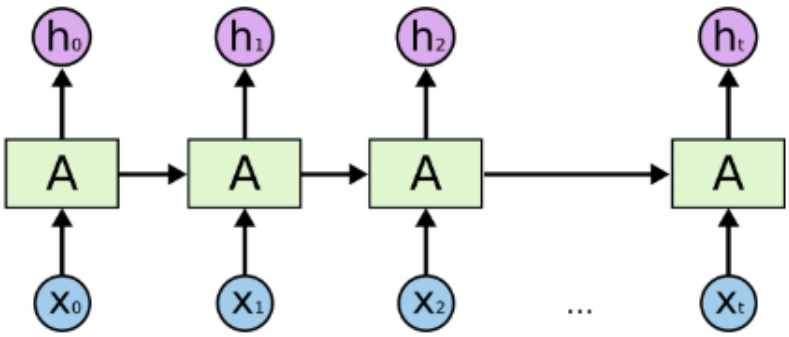

Figura 2.1: Fluxo básico da informação em uma RNN.

As redes *GRU (Gated Recurrent Unit)* e *LSTM (Long Short-Term Memory)* são dois exemplos de redes recorrentes conhecidas [16]. As *LSTM* são um tipo especial de rede recorrente projetado para lidar com o problema do "desaparecimento do gradiente", que é comum em rede recorrente tradicionais como dito anteriormente [24]. Esse problema impede que as redes recorrentes capturem dependências de longo prazo em sequências de dados. As *LSTM* melhoram isso utilizando uma arquitetura mais complexa que permite a retenção de informações por períodos mais longos [24]. A figura 2.2 mostra a estrutura de uma célula *LSTM* que possui uma arquitetura bem mais sofisticada que a rede neural recorrente padrão.

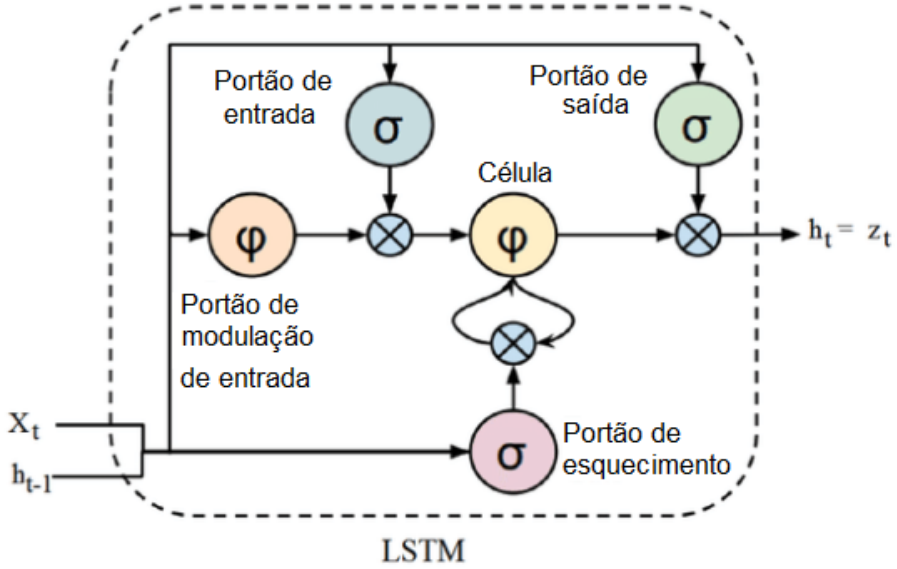

Figura 2.2: Estrutura básica de uma célula LSTM.

2.3 Processamento de Linguagem Natural

A área do PLN [22] situa-se na interseção entre a linguística computacional e a inteligência artificial [25]. Seu principal objetivo é buscar o desenvolvimento de modelos e sistemas que capacitem os computadores a compreender, interpretar, processar e gerar texto ou fala de forma análoga à capacidade humana.

Por meio do PLN, busca-se não apenas a análise superficial da linguagem mas também uma compreensão mais profunda do significado, contexto e intenção por trás das expressões linguísticas, capacitando computadores a realizar uma melhor manipulação da linguagem humana, visando atingir uma comunicação de maneira semelhante a que os seres humanos fazem.

Vale ressaltar o fato de que os computadores são incapazes de processar e entender textos de forma direta, sendo necessário converter textos e palavras em representações

vetoriais que possam ser processadas pelos computadores. Essas representações vetoriais são amplamente utilizadas em inteligência artificial e aprendizado de máquina e são chamadas de *embeddings*. Para essa tarefa, há alguns anos, foram criadas técnicas como Bag Of Words, TF-IDF e Word2Vec [7]. Atualmente, tem-se técnicas mais avançadas como o ELMo [41] e BERT [12]. Essas técnicas permitem que dados complexos como palavras, imagens ou outros tipos de informação, sejam transformados em vetores de números, permitindo a análise e o processamento por algoritmos de aprendizado de máquina e modelos de inteligência artificial.

A natureza da linguagem humana é complexa, pois envolve nuances semânticas, gramaticais, sintáticas [22]. Portanto, segundo Khurana e outros [25], o PLN aborda uma série de desafios técnicos, entre eles:

1. Análise léxica: este processo permite identificar se as palavras utilizadas em um texto são válidas, isto é, possuem significado na língua em questão.
2. Análise sintática: processo de identificação de que a frase está estruturada de forma correta ou não, bem como a classificação sintática de cada elemento que integram uma frase.
3. Análise semântica: visa compreender o significado das palavras e da frase formada por elas. Para isso, leva-se em consideração o contexto e a intenção comunicativa. Essa ação pode envolver a identificação de sinônimos, antônimos, polissemia e ambiguidade.
4. Processamento morfológico: trata-se da análise da forma e estrutura das palavras, incluindo flexões, prefixos, sufixos e outras alterações morfológicas que afetam o significado das palavras.
5. Ambiguidade: as linguagens naturais criadas pelos humanos ao longo do tempo são, intrinsecamente ambíguas, apresentando múltiplas interpretações possíveis para uma mesma sentença. Isso pode incluir tanto as ambiguidades semânticas (palavras e frases com múltiplos significados), como as ambiguidades sintáticas que são o uso de diferentes estruturas gramaticais para expressar uma mesma ideia.
6. Variabilidade linguística: a linguagem varia amplamente entre diferentes contextos sociais, culturais e individuais. Diferenças de sotaque, gírias, jargões e variações dialetais representam um desafio para os sistemas de PLN, que precisam ser capazes de lidar com essa diversidade linguística.
7. Contextualização: a compreensão adequada da linguagem requer a consideração do contexto em que as palavras ou sentenças são utilizadas. Isso inclui inferir significados a partir de conhecimento prévio, contexto histórico e informações situacionais.
8. Dependências de longa distância: muitas vezes, o significado de uma palavra ou frase depende de informações que estão distantes na sentença ou no texto.

Identificar e capturar essas dependências de longa distância é um desafio para os sistemas de PLN.

Esses são apenas alguns dos muitos aspectos a serem tratados pelas tarefas dentro do PLN. Para enfrentar esses desafios, os pesquisadores desenvolveram uma variedade de técnicas e algoritmos, incluindo abordagens baseadas em regras estatísticas e aprendizado de máquina. Mais recentemente, o uso de redes neurais profundas e modelos de linguagem pré-treinados [22] têm apresentado melhores resultado que as abordagens anteriores.

Segundo Jusoh [23], o PLN tem uma ampla gama de aplicações práticas em diferentes setores, como por exemplo:

- Assistência virtual e chatbots.
- Análise de sentimento.
- Reconhecimento de fala.
- Sumarização automática de texto.
- Tradução automática.
- Reconhecimento de Entidades Nomeadas.
- Motores de busca e sistemas de recomendação.

Um termo bastante comum em PLN e que será citado em vários pontos deste trabalho, é o *token*. Um token é uma unidade básica de texto que é usada para análise e processamento. Um token pode ser uma palavra, parte de uma palavra, ou até mesmo um símbolo. O processo de dividir um texto em tokens é chamado de tokenização. Esse processo se divisão pode ser feito de várias formas diferentes, o algoritmo pode fazer a divisão por palavras, por caracteres ou até mesmo dividir as palavras em trios de caracteres. Dado o exemplo de texto a seguir: "O número do meu telefone é (62) 9911-9911.", o tokens gerados poderiam ser: "o", "numero", "do", "meu", "telefone", "e", "62", "9911", "9911", ".".

Segue nas próximas seções um detalhamento de recursos que foram desenvolvidos ao longo do tempo que permitiu que o PLN chegasse ao estágio atual.

2.4 Modelo de Linguagem

A Era dos Modelos de Linguagem Pré-Treinados (década de 2010 em diante) produziu diversos trabalhos que evoluíram velozmente os recursos de PLN. Dentre esses trabalhos pode-se citar [54] que alavancou o uso das redes LSTM e o trabalho de Mikolov e Zweig [37] que propôs um modelo inicial de linguagem utilizando as redes LSTM.

Nos últimos anos, assistimos a uma revolução nos modelos de linguagem com o surgimento de modelos pré-treinados massivos [25]. Esses Modelos pré-treinados apresentam um desempenho excepcional em uma variedade de tarefas de PLN, permitindo

a criação de aplicações computacionais que até recentemente eram inimagináveis dada as dificuldades que os modelos tradicionais apresentavam para realizá-las.

Os modelos de linguagem pré-treinados são sistemas de inteligência artificial, treinados em grandes quantidades de texto não rotulado antes de serem ajustados para tarefas específicas de PLN [69]. Durante o treinamento, o modelo aprende a capturar padrões complexos e estruturas linguísticas presentes nos dados, desenvolvendo uma representação interna rica e generalizada da linguagem.

Após o treinamento inicial em dados não supervisionados, os modelos de linguagem pré-treinados podem ser ajustados (*fine-tuning* ou ajuste fino) para tarefas específicas de PLN, como classificação de texto, tradução automática, resumo de texto, questionamento e resposta, entre outros [22].

O ajuste fino é de certa forma semelhante ao treinamento, mas de uma forma reduzida e específica. Ele, ao contrário de fazer o modelo aprender um grande contexto como é o caso de uma linguagem, o modelo é treinado em um conjunto de dados bem menor com o objetivo de se especializar em resolver uma tarefa específica. Segundo [52], o modelo é treinado em um conjunto de dados rotulado para aprender a realizar a tarefa desejada de forma mais eficaz. A ideia de reaproveitar o conhecimento dos modelos pré-treinados para as tarefas especificas é chamado de transferência de aprendizado.

Os modelos pré-treinados têm sido amplamente adotados na indústria e na pesquisa de PLN devido à sua capacidade de fornecer representações de linguagem de alta qualidade e desempenho superior em uma variedade de tarefas de processamento de texto [60]. Eles permitem uma abordagem mais eficiente e generalizada para o desenvolvimento de sistemas de PLN, reduzindo a necessidade de treinamento supervisionado específico para cada tarefa e aproveitando o conhecimento prévio aprendido [33] durante o treinamento em grandes quantidades de texto não rotulado.

Existem vários modelos de linguagem pré-treinados amplamente utilizados, cada um com suas próprias arquiteturas e métodos de treinamento. Alguns dos modelos mais conhecidos incluem:

- ELMo (*Embeddings from Language Model*) [41]: é um dos primeiros modelos de linguagem, desenvolvido em 2017. Ele trouxe a ideia inovadora de embeddings contextuais. Ao pré-treinar em grandes conjuntos de dados, o ELMo capturou nuances semânticas, fornecendo uma base para modelos subsequentes.
- BERT (*Bidirectional Encoder Representations from Transformers*) [12]: introduzido pelo Google AI em 2018, o BERT é um modelo de linguagem baseado em uma arquitetura Transformers. Ele utiliza a ideia de bidirecionalidade no pré-treinamento de modelos de linguagem entre outras técnicas durante o treinamento. Essa abordagem melhorou significativamente o desempenho em uma variedade de tarefas.

- GPT (*Generative Pre-trained Transformer*) [42]: desenvolvido pela OpenAI. Esse modelo pré-treinado em grandes quantidades de texto conseguiu gerar texto de forma coerente e relevante. O GPT utiliza uma arquitetura de Transformer.
- RoBERTa (*Robustly Optimized BERT Approach*): é uma versão do BERT que utiliza pré-processamento e hiperparâmetros otimizados para melhorar o desempenho do modelo.
- XLNet (*eXtreme Language Understanding Network*) [64]: introduzido pelo *Google Research*, o XLNet usa uma abordagem autoregressiva e permutação de *tokens* para capturar dependências de longo alcance em sequências de texto. Ele combina elementos do BERT e do *Transformer-XL* [9], introduzindo uma perspectiva autorregressiva enquanto captura informações bidirecionais.

2.5 Ajuste fino

Apesar do relevante avanço no estado da arte de processamento de linguagem natural, a maioria das aplicações no setor industrial depende de uma grande quantidade de dados para treinamento dos modelos de linguagem de grande porte. Assim, passaram a buscar formas de adaptar modelos de linguagem para aplicações específicas de modo a consumir uma baixa quantidade de dados e, consequentemente, uma menor demanda de capacidade computacional. Esse tipo de adaptação frequentemente é chamado de ajuste fino (fine-tuning) [52].

Os modelos pré-treinados têm mostrado grande sucesso em melhorar o desempenho em várias tarefas, entretanto ainda é necessário um ajuste fino para a tarefa. E, assim como em vários outros problemas de *deep learning*, é necessário um conjunto de dados rotulado para fazer o ajuste fino.

Um tipo de ajuste fino clássico é o ajuste fino completo, que basicamente atualiza todos os parâmetros do modelo ao especializa-lo em uma tarefa específica. Embora demonstre certa eficácia, fazer o ajuste fino completo traz um conjunto de desafios e problemas. Quanto maior o modelo, mais parâmetros para ajustar, logo mais memória, processamento e dificuldade para se encontrar o ponto ótimo das alterações. Encontrar a quantidade ideal de épocas, taxa de aprendizado e quantidade de dados a serem utilizados para se fazer um ajuste fino completo eficiente pode ser um processo bem trabalhoso. Caso o ajuste fino completo faça alterações excessiva nos parâmetros, o modelo original pode começar a esquecer conhecimento adquirido, piorando os resultados do aprendizado [52].

O ajuste fino é feito para tarefas ou contextos mais específicos, portanto, dificilmente poderá ser reaproveitado para outras tarefas. Dessa forma, para cada tarefa diferente, um novo ajuste fino precisará ser feito.

Para superar alguns dos problemas do ajuste fino completo foram desenvolvidas ao longo do tempo técnicas mais eficientes, conhecidas como técnicas ajuste fino com eficiência de parâmetros, tais técnicas serão detalhadas mais adiante nesse trabalho.

2.6 Arquitetura Transformers

O artigo "*Attention Is All You Need*"de Matthew e outros [58] revolucionou a área do Processamento de Linguagem Natural e da aprendizagem de máquina, apresentando a arquitetura de *Transformer*. Trata-se de um modelo que dispensa completamente a utilização de arquiteturas recorrentes ou convolucionais em favor de mecanismos puramente baseados em atenção [58]. Publicado em 2017, o trabalho introduz um novo paradigma para a construção de modelos de linguagem, alcançando resultados de desempenho excepcionais em uma variedade de tarefas.

O *Transformer* representa uma mudança de paradigma significativa em relação aos modelos anteriores, como os baseados em redes neurais recorrentes (RNNs) [51] e redes neurais convolucionais (CNNs) [38]. Ao contrário das abordagens que dependem de informações sequenciais ou de convoluções locais para processar dados, o *Transformer* opera utilizando o mecanismo de atenção [58]. Esse mecanismo permite ao modelo atribuir pesos diferentes a diferentes partes da entrada, permitindo uma aprendizagem mais eficiente e paralela [58].

Uma das principais inovações introduzidas pelo *Transformer* é a atenção multi-cabeça, que permite ao modelo focalizar simultaneamente em diferentes partes da entrada em múltiplas perspectivas. Isso não só melhora a representação dos dados, mas também permite ao modelo capturar relações mais complexas entre as palavras em uma sentença. Além disso, o *Transformer* introduz o conceito de codificadores e decodificadores [58], onde cada *token* em uma sequência é processado considerando toda a sequência, em vez de apenas as informações locais.

Outro aspecto crucial do *Transformer* é a sua capacidade de lidar com sequências de comprimento variável de forma eficiente, no sentido de capturar a informação contida, graças ao uso de mecanismos de atenção que não dependem do comprimento da sequência. Isso torna o modelo especialmente adequado para tarefas que envolvem sequências longas, como tradução de texto e modelagem de linguagem.

Além da arquitetura *Transformer* em si, o trabalho de Matthew e outros [58] também fez uso de técnicas de treinamento eficazes para o Transformer, incluindo a utilização de otimização baseada em mini-lotes e o *dropout*. O *dropout* é uma técnica de regularização usada para prevenir *overfitting* em redes neurais, em cada iteração de treinamento, cada neurônio (excluindo os da camada de saída) tem uma probabilidade de ser desligado, os neurônios que são desligados não participam no processo de *feedforward*

e *backpropagation* dessa iteração específica. Essas técnicas são essenciais para garantir o bom desempenho do modelo em uma variedade de tarefas e conjuntos de dados.

Desde a sua publicação, o *Transformer* tem sido amplamente adotado pela comunidade de pesquisa em aprendizado de máquina e processamento de linguagem natural. Ele tornou-se a base para uma série de modelos de estado-da-arte em uma variedade de aplicações, incluindo tradução automática, sumarização de texto, geração de texto e muito mais. Sua arquitetura flexível e eficiente continua a inspirar novas pesquisas e avanços no campo, consolidando seu status como um marco fundamental na história do aprendizado de máquina. A Figura 2.3 ilustra a arquitetura *transformer*. As subseções seguintes detalham os principais blocos funcionais da Figura 2.3.

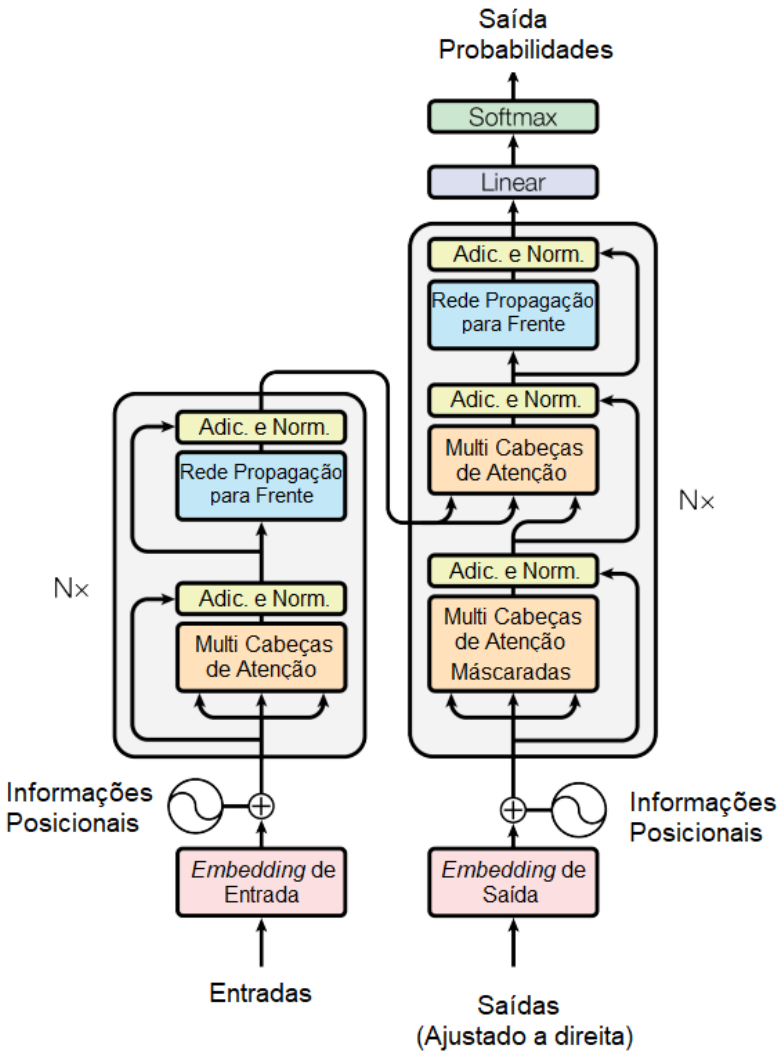

Figura 2.3: Visão Geral da arquitetura *Transformer*.

2.6.1 Camada de Atenção

A atenção é o núcleo da arquitetura *Transformer*. Ela permite que o modelo atribua pesos dinâmicos a cada *token* em uma sequência com base em sua relevância para os outros *tokens* da sequência. Assim, em vez de processar cada *token* de forma sequencial, a atenção permite que o modelo processe todos os *tokens* simultaneamente, capturando dependências entre *tokens* em toda a sequência. A atenção é calculada através de produtos escalares entre as matrizes de *Query*, *Key* e *Value*, resultando em uma matriz de pesos de atenção que é aplicada aos valores para calcular as representações finais. No decorrer do treinamento as várias cabeças de atenção irão capturar diferentes relações entre os tokens, conforme ilustra a figura 2.4 abaixo.

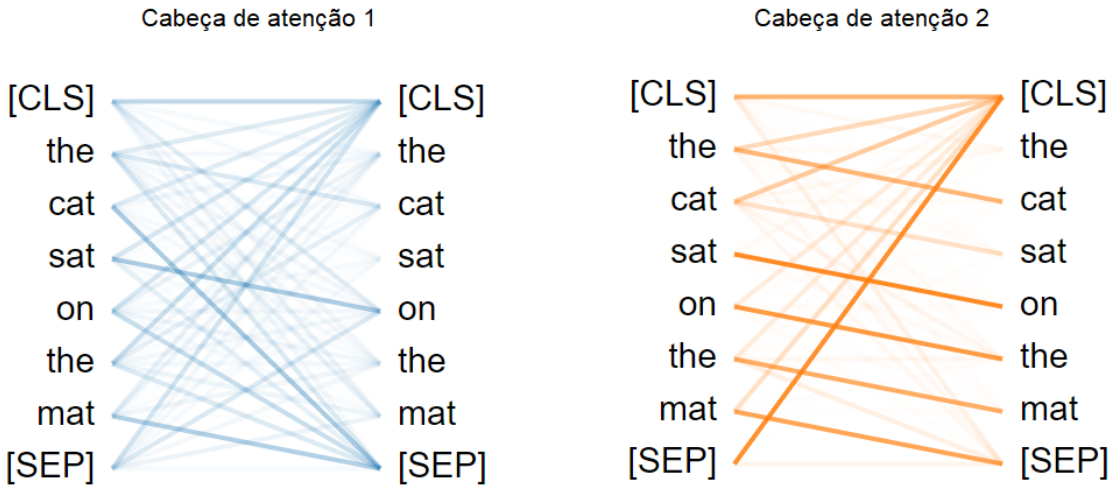

Figura 2.4: Relação entre tokens em duas cabeças de atenção.

2.6.2 Conexões Residuais

Para facilitar o treinamento de redes profundas, a cada subcamada (atenção e *feedforward*) é adicionada a uma conexão residual antes da normalização. Essas conexões residuais ajudam a mitigar o problema de desaparecimento de gradiente em redes profundas, permitindo que os gradientes fluam com valores mais significativos durante a fase de *backpropagation*[58].

2.6.3 Camada de Normalização

Após o cálculo da atenção, as saídas são normalizadas para evitar problemas como degradação do gradiente durante o treinamento. A normalização é aplicada em

cada subcamada (atenção e *feedforward*) e é essencial para garantir que o treinamento seja estável e eficaz.

2.6.4 Camada *Feedforward*

Após o cálculo da atenção e da normalização, as representações dos *tokens* passam pela camada *feedforward*. Essa camada é composta de redes neurais densas, geralmente com ativação *ReLU* [26], que transformam as representações dos *tokens* em espaços de maior dimensionalidade. Isso ajuda o modelo a aprender representações mais complexas e não-lineares das sequências de entrada [58].

2.6.5 Codificador e Decodificador

Os modelos *Transformer* são frequentemente organizados em uma arquitetura de codificador-decodificador. O codificador recebe a sequência de entrada e gera representações contextualizadas dos *tokens*. Já o decodificador, por sua vez, recebe as representações do codificador e as usa para gerar sequências de saída, como tradução de texto em sistemas de tradução automática.

2.7 BERT

O BERT (*Bidirectional Encoder Representations from Transformers*) [12] é baseado na arquitetura *Transformer*. Essa abordagem revolucionou o PLN com sua capacidade de capturar, de forma eficiente, dependências de longo alcance em sequências textuais. No entanto, ao contrário dos modelos *Transformer* anteriores que eram treinados de forma unidirecional, o BERT propõe um treinamento bidirecional, permitindo que o modelo capture contextos de palavras de ambos os lados em uma sequência.

Esta inovação no processo de treinamento do modelo BERT [12] é uma parte fundamental de seu sucesso em alcançar desempenho de estado-da-arte em uma variedade de tarefas de PLN. A Figura 2.7 mostra uma visão geral da arquitetura do processo de treinamento do BERT. Segue um breve detalhamentos dos passos que compõem o processo.

2.7.1 Pré-Processamento de Dados

Antes do treinamento, os dados são pré-processados para se adequarem à entrada do modelo BERT. Esse processo inclui a tokenização, a segmentação de sentenças e a adição de *tokens* especiais, como [CLS] (*token* de início de sentença) e [SEP] (separador de sentença). Tarefas de Pré-Treinamento:

O BERT é pré-treinado em duas tarefas principais:

Preenchimento de Lacunas (*Masked Language Model - MLM*)

Cerca de 15% dos *tokens* de entrada são mascarados aleatoriamente, substituídos pelo *token* especial "[MASK]". O modelo é então treinado para prever as palavras mascaradas com base no contexto fornecido pelas outras palavras na sequência. A Figura 2.7 mostra um exemplo onde uma palavra da sentença é mascarado e a partir disso o modelo deve prever qual a palavra correta.

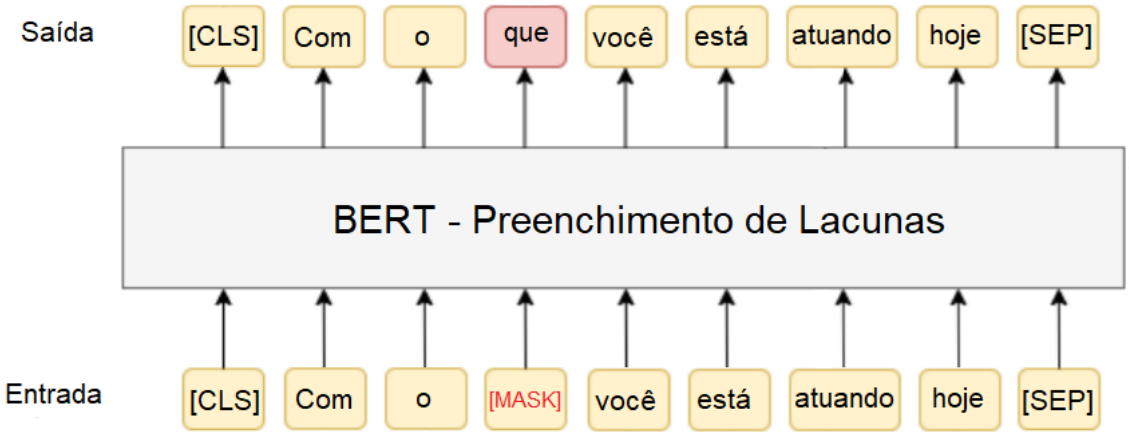

Figura 2.5: Fluxo de aplicação de máscara e previsão do token no BERT.

Previsão da Próxima Sentença - *Next Sentence Prediction (NSP)*

Nesta etapa, o modelo recebe pares de sentenças e é treinado para prever se a segunda sentença é uma continuação lógica da primeira ou não. Vários pares de sentenças são retirados do corpus de treinamento, e também são gerados pares aleatórios onde a segunda sentença não é uma continuação lógica da primeira. A Figura 2.6 mostra como é estruturado um par de sentença enviado para o BERT. Na saída do modelo o token [CLS] terá a informação binária (sim ou não) se o par sentença é uma continuação lógica ou não.

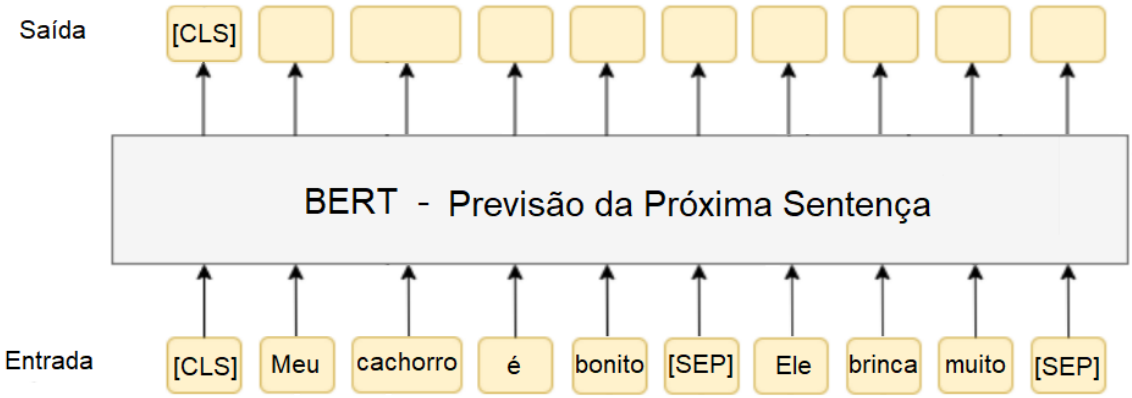

Figura 2.6: Fluxo de previsão de próxima sentença no BERT.

2.7.2 Estrutura de Entrada

Os *tokens* de entrada são representados por embeddings de *token*, que são somados aos embeddings posicionais e embeddings de segmentos, conforme é ilustrado pela figura 2.7. Essa sequência de embeddings é então passada para a pilha de camadas de codificação *Transformer* [12]. Os embeddings posicionais contém a informação da posição sequencial de cada token, a ideia é fazer com que o modelo faça uso de informação posicional que agora está contida em cada um dos tokens. Já os embeddings de segmentos são um mecanismo que permite ao modelo distinguir entre diferentes partes de uma sequência de entrada, especialmente útil para tarefas que envolvem pares de sentenças, como a previsão da próxima sentença.

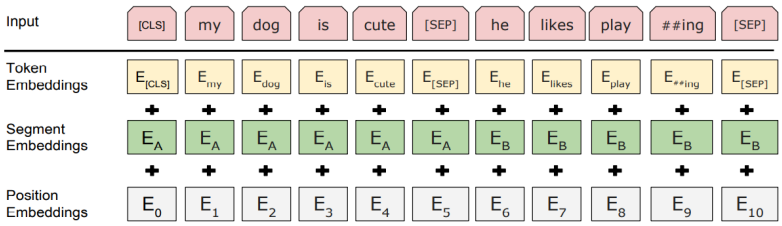

Figura 2.7: Visão geral da estrutura de entrada de dados no BERT.

2.7.3 Codificação Bidirecional

Cada camada de codificação *Transformer* no BERT, usa a abordagem de atenção multi-cabeça para permitir que o modelo capture informações de contexto de palavras de ambos os lados da sequência[12]. Isso é crucial para garantir que o modelo tenha uma compreensão abrangente do contexto em torno de cada palavra.

2.7.4 Ajuste fino do BERT para classificação de texto

Os conjuntos de dados e os ajustes finos feitos neste trabalho estão diretamente relacionados à tarefa de classificação de texto. O ajuste fino do modelo BERT para a tarefa de classificação de texto envolve adaptar o modelo pré-treinado em uma grande quantidade de dados não rotulados. Esse ajuste permite a realização de tarefas específicas de classificação de texto em categorias ou rótulos predefinidos.

A visão da arquitetura do processo de ajuste fino para classificação de texto no BERT é ilustrado na Figura 2.8.

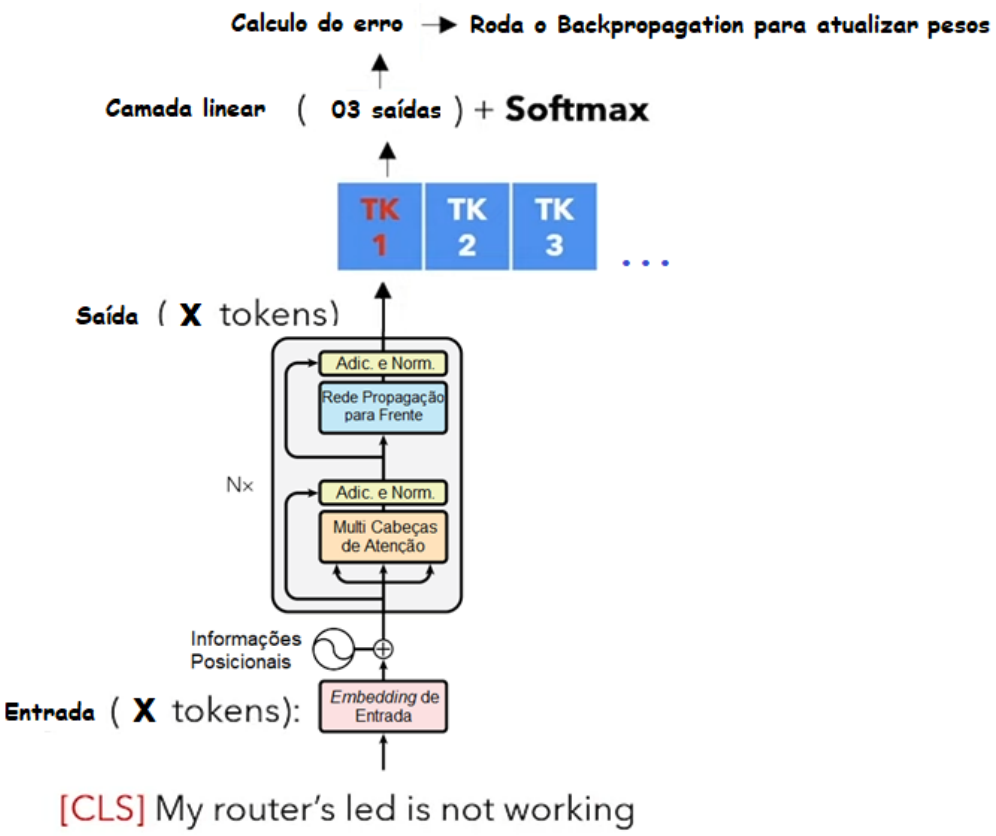

Figura 2.8: Processo de classificação no ambiente do BERT.

O processo é feito utilizando somente a parte do *encoder* do *transformer*, com as seguintes epatas:

1. A sentença a ser classificada é tokenizada e transformada em embeddings seguindo o padrão do BERT.
2. A sentença segue todo o fluxo da arquitetura *transformer* contida no BERT até sua saída.
3. Vale lembrar que o primeiro *token* da sentença do BERT é o *token* padrão [CLS], tanto na entrada, quanto na saída. E, são os embeddings desse *token* que são

enviados para uma camada linear que possui saídas produzidas por uma função de ativação softmax, onde cada saída é uma das possíveis classificações de texto.

4. A partir dessa saída é calculada o erro e é executado o *backpropagation* atualizando os pesos do modelo.

2.8 DistilBERT

O DistilBERT [48] é uma versão compacta e mais leve do BERT, desenvolvida pela equipe da *Hugging Face* em 2019 [20], que passou pelo processo de destilação (*knowledge distillation*) durante o seu processo de treinamento. O processo de destilação, é uma técnica onde um modelo maior (conhecido como "teacher") é usado para treinar um modelo menor e mais simples (conhecido como "student") [15] [48]. O objetivo é transferir o conhecimento do modelo maior para o modelo menor, de forma que o modelo menor possa alcançar o desempenho mais próximo possível do modelo maior. A figura 2.9 demonstra o ciclo do processo de destilação.

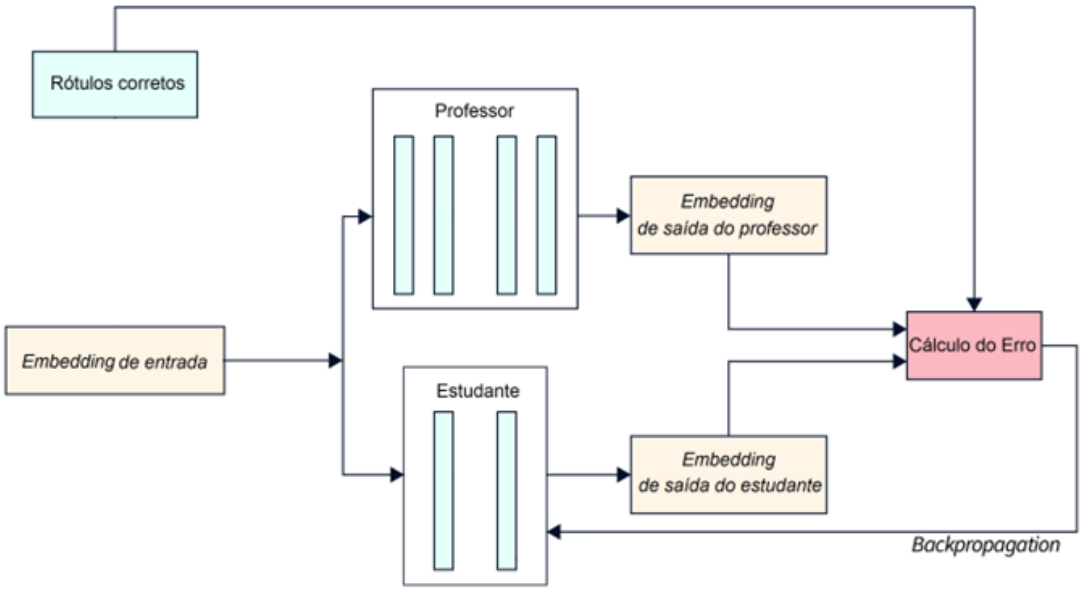

Figura 2.9: Ciclo do processo de destilação.

Durante a fase de distillation o processo de treinamento do modelo menor (student) tem como objetivo faze-lo imitar o comportamento do modelo maior (teacher), isso é feito utilizando uma combinação de duas funções de perda que serão explicadas dos dois tópicos abaixo.

2.8.1 *Distillation loss*

O modelo teacher é usado para gerar previsões sobre os dados de treinamento e em vez de usar apenas as classes de rótulos, o modelo teacher gera distribuições de probabilidade sobre todas as classes, essas distribuições de probabilidade fornecem informações adicionais sobre as incertezas do modelo e as relações entre as diferentes classes. O modelo student faz a mesma predição gerando uma diferente distribuição de probabilidades, a perda é calculada pela diferença da previsão do modelo student para o modelo teacher. E então via backpropagation os parâmetros do modelo student são atualizados para que suas previsões se aproximem do modelo teacher [15] [48].

2.8.2 *Classic loss*

É basicamente um treinamento convencional de modelos, a ideia é que o modelo student também esteja aprendendo diretamente dos dados rotulados. O conjunto de dados é apresentado ao modelo student, e com base no erro da previsão os parâmetros do modelo student são atualizados via backpropagation [15] [48].

2.8.3 DistilBERT vs BERT

Embora ambos os modelos são baseados na arquitetura *Transformer* e são usados para uma variedade de tarefas de PLN, existem algumas diferenças-chave entre o BERT e o DistilBERT, tais como:

1. O BERT-base original é um modelo grande, com 110 milhões de parâmetros, enquanto o DistilBERT é uma versão reduzida do BERT, com apenas 6 camadas de codificação e cerca de 66 milhões de parâmetros.
2. O DistilBERT foi projetado para ser mais leve e eficiente em termos de computação, ocupando menos espaço em memória e exigindo menos recursos computacionais durante a inferência, tornando-o mais adequado para implantações em dispositivos com recursos limitados.
3. Devido ao seu tamanho reduzido, o DistilBERT tende a ser mais rápido que o BERT original em termos de processamento durante a inferência . Isso o torna uma escolha atraente para aplicativos que exigem baixa latência ou que são executados em dispositivos com recursos limitados, como dispositivos móveis ou sistemas embarcados.
4. Embora o DistilBERT seja uma versão compacta do BERT, ele ainda mantém um desempenho notavelmente bom em uma variedade de tarefas de PLN. Em alguns casos, o DistilBERT se aproxima do desempenho do BERT original em tarefas como em classificação de texto, resposta a perguntas e sumarização de texto.

5. O DistilBERT é pré-treinado de maneira semelhante ao BERT, utilizando uma combinação de preenchimento de lacunas (MLM) e previsão da próxima sentença (NSP) em grandes corpora de texto não rotulado.

2.9 RoBERTA

O RoBERTa *(Robustly optimized BERT approach)* [31] é uma variação do BERT que foi proposta pela equipe do *Facebook AI* em 2019 [3]. Os autores do RoBERTa fizeram um estudo de replicação do pré-treinamento do BERT que inclue uma avaliação cuidadosa dos efeitos dos hiperparâmetros e do tamanho do conjunto de dados do treinamento. O trabalho concluiu que o BERT foi subtreinado e propuseram uma serie de mudanças no treinamento [31]. Entre as várias sugestões dadas pelo trabalho, segue quatro das mais importantes:

1. Treinar o modelo por mais tempo, com lotes maiores e com mais dados.
2. Não utilizar a tarefa *Predição da próxima sentença* ou *Next Sentence Prediction* durante o treinamento.
3. Treinamento em sequências mais longas.
4. Alterar dinamicamente o padrão de mascaramento aplicado aos dados de treinamento.

O trabalho comparou o RoBERTa com o BERT-Large, nos conjuntos de dados SQuAD, MNLI-m, SST-2 e GLUE, tendo o RoBERTa melhores em todos os testes, e com isso o trabalho denominou o RoBERTa como uma versão mais robusta e otimizada do BERT.

2.10 Grandes Modelos de Linguagem

Large Language Models(LLM) [69], ou Modelos de Linguagem Grandes, referem-se a uma classe de modelos de Linguagem que são caracterizados por sua escala massiva em termos de quantidade de parâmetros e necessidade de capacidade computacional. Esses modelos têm demonstrado desempenho excepcional em uma variedade de tarefas [69].

Os LLMs, assim como os LM, como o BERT, por exemplo, são construídos baseados em arquiteturas de redes neurais profundas, como os modelos *Transformers*, que são capazes de lidar com sequências longas de texto e capturar dependências complexas entre palavras. LLMs exibem fortes capacidades para compreender a linguagem natural e resolver tarefas complexas (via geração de texto) [68].

Os LLMs se destacam por algumas características como:

1. Escalabilidade: esses modelos são projetados para lidar com grandes quantidades de dados e parâmetros. Eles são, necessariamente, treinados em supercomputadores ou infraestrutura de computação distribuída para processar e armazenar enormes conjuntos de dados de texto.
2. Generalização: Os LLMs são treinados em uma ampla gama de tarefas linguísticas e domínios de texto, o que lhes conferem uma capacidade excepcional de generalização. Eles podem capturar padrões linguísticos complexos e se adaptarem a diferentes contextos linguísticos.

Junto com as LLM's veio também a possibilidade de se utilizar prompts [47], que de certa forma é uma evolução do que já havia sido oferecido nos modelos T5 [45]. O prompt é uma das alternativas ao ajuste fino com algumas vantagens (e desvantagens em vários aspectos também) como não ter que envolver programação ou retreinamento do modelo [52]. A qualidade do prompt pode ter grande impacto na resposta do modelo. A seguir um simples exemplo de prompt que poderia ser enviado a uma LLM: "O sentimento da seguinte frase é positivo ou negativo? Frase: O filme foi excelente". Este prompt de maneira clara informa o modelo o que deve ser feito com a frase enviada.

A Engenharia de Prompts [52] é o processo da criação de forma cuidadosa de prompts. Consiste em criar um conjunto de instruções que são fornecidas a um LLM no momento da que, de certa forma, "reprograma"o LLM, melhorando suas capacidades de responder de forma mais precisa.

A criação do prompt pode levar em conta uma série aspectos como regras, filtros, limitação do contexto, dados sobre quais informações são importantes e sobre como a saída desejada deve ser formatada [52]. Ao elaborar um prompt, pode-se considerar qual o público-alvo que irá receber a resposta, detalhando ou resumindo a saída de acordo com o perfil do leitor.

Os prompts podem ser projetados para programar um LLM para fazer coisas bem avançadas e não apenas guias e filtros básicos. Por exemplo, tarefas que o prompt pode especificar para direcionar a resposta da LLM:

- Gerar somente códigos seguindo um paradigma de programação.
- Simular um terminal Linux.
- Criar uma prova com várias perguntas sobre conceitos de engenharia de software.
- Estabelecer saídas customizadas ao contexto que o leitor está habituado.
- Ressignificar a semântica de palavras para atender contextos regionais.

2.11 Llama

O Llama [55] é uma coleção de modelos de linguagem que variam de 7 bilhões a 70 bilhões de parâmetros, lançada pela Meta AI em 2023 [3]. O modelo foi treinado utilizando trilhões de *tokens*, sendo que uma das motivações era mostrar que é possível treinar modelos de última geração usando exclusivamente conjuntos de dados disponíveis publicamente, sem recorrer a conjuntos de dados proprietários.

De acordo com o Touvron e outros [55], o LLaMA-13B supera GPT-3 (175B) na maioria dos *benchmarks* e é competitivo com os outros bons modelos maiores como o Chinchila-70B e PaLM-540B. Seguindo trabalhos recentes sobre grandes modelos de linguagem, o Llama é baseada na arquitetura do *transformer*. Além disso, os autores também se aproveitaram de algumas mudanças que foram propostas por outros trabalhos, como o PaLM [2].

Segue algumas das principais alterações em cima da arquitetura *Transformer* e a fonte da motivação da mudança:

1. Pré-normalização [GPT3]. Para melhorar o estabilidade de treinamento, foi normalizado a entrada de cada subcamada do *Transformer*, em vez de normalizar a saída. Foi usada a função de normalização RMSNorm, introduzida por Zhang e Sennrich (2019) [50].
2. Função de ativação SwiGLU [PaLM]. Foi substituída a não linearidade *ReLU* pela função de ativação *SwiGLU*, introduzida por Shazeer (2020) [45] para melhorar o desempenho.
3. *Embeddings* Rotativos [GPTNeo]. Foi removido os *Embeddings* posicionais absolutas e no lugar adicionado embeddings posicionais rotativos (RoPE), introduzidos por [53], em cada camada da rede.

2.12 Tiny Llama

O TinyLlama [65] é um modelo de linguagem de 1.1B de parâmetros pré-treinado em cerca de 1 trilhão de *tokens* por aproximadamente 3 épocas. Com arquitetura baseada na do Llama 2, tendo como diferença o uso de uma quantidade menor de parâmetros, e também fazendo uso do tokenizador do Llama 2, o TinyLlama se aproveita de alguns avanços feitos pela comunidade de código aberto como o FlashAttention [10], para alcançar melhor eficiência computacional. Apesar de seu tamanho relativamente pequeno, o TinyLlama demonstra desempenho notável em uma série de tarefas superando outros modelos de linguagem de código aberto existentes com tamanhos comparáveis [65].

2.13 Técnicas para redução de modelo

Existem diversas técnicas que tentam reduzir o tamanho do modelo seja durante o treinamento, ajuste fino ou inferência, podemos citar algumas bem conhecidas como o *prunning* (poda), *quantization* (quantização) e *Parameter-Efficient Fine-Tuning* (Ajuste fino com eficiência de parâmetros). Este trabalho descreve com detalhes as técnicas de ajuste fino com eficiência de parâmetros, então neste tópico será abordado de forma breve o apenas o *prunning* e *quantization*.

2.13.1 Prunning

São técnicas usadas para reduzir o tamanho e a complexidade de um modelo. O objetivo principal do *prunning* é simplificar o modelo, diminuindo o número de parâmetros ou a carga computacional exigida, e com isso melhorar a eficiência e reduzindo os requisitos de recursos computacionais, sem comprometer significativamente o desempenho [57]. Existem diversas técnicas para se realizar o *prunning*, podemos citar aqui algumas das principais:

- *Magnitude-Based Pruning*: Remove pesos com valores muito pequenos, assumindo que eles têm um impacto menor sobre as predições.
- *Structured Pruning*: Remove grupos de neurônios ou filtros inteiros, o que facilita a implementação e a aceleração em hardware.
- *Random Pruning*: Remove parâmetros de maneira aleatória. É uma abordagem mais simples, geralmente menos eficiente do que métodos baseados em importância.

2.13.2 Quantization

Quantização de modelos são técnicas que reduz a precisão dos números usados para representar os parâmetros de um modelo, com o objetivo de diminuir o uso de memória e acelerar o tempo de inferência. Isso é feito ao converter os valores dos pesos de uma precisão maior, como 32 bits em ponto flutuante, para uma precisão menor como 8 bits inteiros, reduzindo assim o tamanho do modelo e a carga computacional [46].

Existem diversas vantagens em quantizar um modelo como redução de tamanho, aceleração da inferência e menor consumo de energia. Mas também há desvantagens como perda de precisão o que pode fazer o modelo ter resultados piores e também possível necessidade de um novo treinamento ou ajuste fino, o que aumenta a complexidade do processo como um todo [46].

2.14 Técnicas de Ajuste fino com eficiência de parâmetros

Existem três categorias principais de métodos de ajuste fino com eficiência de parâmetros. Além dessas três, existe mais uma que mescla aspectos das três anteriores [29]. As próximas subseções descreve essas categorias de métodos

2.14.1 Métodos aditivos

A ideia principal por trás dos métodos aditivos é adicionar ao modelo pré-treinado parâmetros ou camadas extras e fazer o treinamento alterando os parâmetros apenas os recursos adicionados. Mesmo com parâmetros adicionais esses métodos alcançam melhorias no tempo e memória do treinamento. [29]. Tais métodos podem utilizar os recursos descritos nos próximas subitens.

Adapter-like

Inicialmente introduzia pequenas redes totalmente conectadas após as subcamadas do *Transformer* [17]. Após isso diversas variações foram propostas modificando o local das camadas adicionadas além de tentativas de reduzir a quantidade de parâmetros treináveis.

Soft prompts

Ao utilizar prompts no modelo de linguagem, se tenta alterar o comportamento de um LLM modificando-o por meio do texto da entrada. Essa técnica demonstrou um desempenho muito bom em cenários de zero e *few-shot*. Por outro lado, utilizar *prompts* é impraticável quando há muitos exemplos de treinamento. Os *prompts* são limitados no número de exemplos de treinamento e pelo tamanho máximo de entrada do modelo [59], [29].

Para superar este desafio, foi proposto o conceito de *soft prompts*, onde uma parte dos *embeddings* de entrada do modelo passa por um ajuste fino via gradiente descendente. Os *soft prompts* podem ser treinados em todas a camadas ou apenas na camada de entrada [29].

2.14.2 Métodos seletivos

As abordagens modernas desses métodos são geralmente baseadas no tipo de camada ou na estrutura interna, como ajustar apenas os *biases* do modelo ou apenas linhas específicas. Outras versões mais agressivas dos métodos seletivos, são os métodos de

atualização esparsa que podem ignorar completamente a estrutura do modelo e selecionar os parâmetros individualmente. As atualizações esparsas de parâmetros apresentam vários desafios de engenharia e eficiência e, ainda, é impraticável em hardware contemporâneo [29].

2.14.3 Métodos baseados em reparametrização

Esse método faz uso de representações de matrizes de dimensão baixa para minimizar o número de parâmetros treináveis. A noção de que as redes neurais têm uma representação dimensional baixa tem sido amplamente explorada na análise empírica e teórica do *deep learning* [29].

Uma série de estudos demonstraram que o ajuste fino pode ser realizado de forma eficaz em subespaços de dimensão baixa [19] [29]. Eles mostraram que o tamanho do subespaço que precisa de adaptação é menor proporcionalmente para modelos maiores. [4].

2.14.4 Métodos Híbridos

Combina ideias de múltiplas categorias de técnicas de Ajuste fino com a eficiência de parâmetros. Os métodos híbridos podem aproveitar os pontos fortes de cada técnica individual ao mesmo tempo que atenuam os seus pontos fracos, levando a um melhor desempenho e eficiência [29].

Abaixo imagem ilustrando várias técnicas existentes de PEFT em seus respectivos grupos.

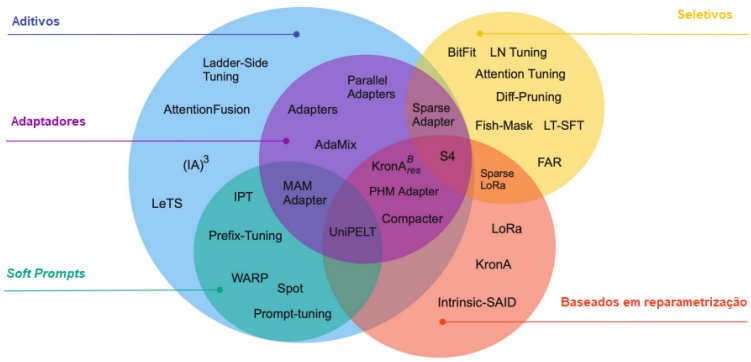

Figura 2.10: Técnicas existentes de PEFT em seus respectivos grupos.
fonte: Imagem retirada de [29].

2.15 Conjuntos de dados

Segue nas próximas subseções, a descrição dos conjuntos de dados utilizados neste trabalho.

2.15.1 IMDb

Conjunto de dados IMDb (Internet Movie Database) com 50 mil resenhas de filmes para processamento de linguagem natural ou análise de texto. É um conjunto de dados para classificação de sentimento binário (positivo ou negativo). O conjunto de dados é dividido em 25.000 resenhas de filmes para treinamento e 25.000 para teste [11]. A Tabela 2.1 apresenta um trecho do conjunto de dados IMDb.

Tabela 2.1: Trecho do conjunto de dados IMDb.

Resenha	Rótulo
One of the other reviewers has mentioned that after watching just 1 Oz episode you'll be hooked. The	positive
I thought this was a wonderful way to spend time on a too hot summer weekend, sitting in the air con...	positive
So im not a big fan of Boll's work but then again not many are. I enjoyed his movie Postal (maybe im...	negative
An awful film! It must have been up against some real stinkers to be nominated for the Golden Globe....	negative

2.15.2 AG News

AG News (AG's News Corpus) é um sub conjunto do corpus de notícias da AG construído pela união de títulos e campos de descrição de artigos das 4 maiores classes. Comumente utilizado em tarefas de classificação de texto, contendo notícias coletadas da web e rotuladas em quatro categorias diferentes: (Mundo, Esportes, Negócios e Ciência/Tecnologia) [67]. O AG News contém 30.000 amostras de treinamento e 1.900 amostras de teste. A Tabela 2.2 apresenta um trecho do conjunto de dados AG News.

Tabela 2.2: Trecho do conjunto de dados AG News.

Resenha	Rótulo
Pope celebrates Mass in Lourdes An ailing Pope John Paul II says Mass at Lourdes, the French shrine revered by Roman Catholics.	World
Colander Misses Chance to Emulate Jones ATHENS (Reuters) - But for a decision that enraged her coach, LaTasha Colander might have been the Marion Jones of the Athens Olympics.	Sports
Wall St. Bears Claw Back Into the Black NEW YORK (Reuters) - Short-sellers, Wall Street's dwindling band of ultra-cynics, are seeing green again.	Business
Sun's Looking Glass Provides 3D View (PC World) PC World - Developers get early code for new operating system 'skin' still being crafted.	Sci/Tech

2.15.3 SNLI

O conjunto de dados SNLI (Stanford Natural Language Inference) consiste em 570 mil pares de frases em inglês escritas e rotuladas por humanos, os rótulos são implicação, contradição e neutro. As premissas são legendas de imagens do conjunto de dados Flickr30k, enquanto as hipóteses foram geradas por anotadores que viram a premissa e foram solicitados a gerar implicações, sentenças contraditórias e neutras [5].

A tarefa principal associada ao SNLI é a inferência textual, que envolve determinar a relação lógica entre duas sentenças: se uma sentença é uma consequência lógica da outra (entailment), se elas são contraditórias (contradiction) ou se são independentes (neutral). Por exemplo, dado o par de sentenças "Um homem está pilotando um avião"e "Um homem está no cockpit de um avião", a relação seria de "entailment", pois a segunda sentença é uma consequência lógica da primeira [5].

A Tabela 2.3 apresenta um trecho do conjunto de dados SNLI.

Tabela 2.3: Trecho do conjunto de dados SNLI.

Premissa	Hipótese	Rótulo
A man inspects the uniform of a figure in some East Asian country.	The man is sleeping	contradiction
An older and younger man smiling.	Two men are smiling and laughing at the cats playing on the floor.	neutral
A black race car starts up in front of a crowd of people.	A man is driving down a lonely road.	contradiction
A soccer game with multiple males playing.	Some men are playing a sport.	entailment
A smiling costumed woman is holding an umbrella.	A happy woman in a fairy costume holds an umbrella.	neutral

2.16 Transferência de aprendizado

O uso de de modelos de linguagem pré-treinado se baseia também no importante conceito de transferência de aprendizado.

A transferência de aprendizado é uma abordagem fundamental no campo de *deep learning*, que se baseia na ideia de aproveitar o conhecimento adquirido em uma tarefa para melhorar o desempenho em outra tarefa relacionada. Essa técnica é especialmente valiosa em cenários onde os conjuntos de dados são limitados ou quando treinar um modelo do zero seria computacionalmente caro. Então, ao invés de iniciar o treinamento de um modelo do zero, a transferência de aprendizado permite que um modelo pré-treinado seja reutilizado para várias outras tarefas especificas, sendo necessário em alguns casos apenas um ajuste fino em um conjunto de dados menor [21] [70].

Modelos pré-treinados em grandes conjuntos de dados tendem a aprender representações gerais úteis dos dados. Ao transferir esse conhecimento para tarefas específicas, os modelos tendem a generalizar melhor [21].

Essa abordagem tem sido amplamente adotada em uma variedade de domínios, desde reconhecimento de imagens e processamento de linguagem natural até tarefas de predição em saúde e finanças, demonstrando sua versatilidade e eficácia em uma ampla gama de cenários de aplicação [21] [70].

2.17 LoRA (*Low-Rank Adaptation*)

O LoRA [19] é uma técnica que acelera o ajuste fino de modelos de linguagem pré-treinados e objetiva reduzir vários dos problemas citados anteriormente para as técnicas de ajuste fino. Se inspirou nos trabalhos [28] [1] que mostra que o aprendizado de modelos, na verdade, residem em uma dimensão intrínseca baixa. Com isso, os autores do LoRA acreditaram que a alteração dos parâmetros durante o ajuste fino também tem uma dimensão intrínseca baixa. O que permite treinar algumas camadas densas da rede neural indiretamente utilizando decomposição de matrizes, enquanto mantém os pesos pré-treinados no modelo original congelados, fazendo do LoRA eficiente na armazenagem e no custo computacional.

2.17.1 Apresentação do Problema

Abaixo uma descrição do problema de modelagem de linguagem, em particular, a maximização de probabilidades condicionais dada uma tarefa específica .

Para um modelo de linguagem pré-treinado $P\phi(y|x)$ parametrizado por ϕ. Então $P\phi(y|x)$ poderia ser um modelo de linguagem pré-treinado apto a aprender várias tarefas específicas como o GPT ([44]; [6]) baseado na arquitetura *Transformer* [58]. Para adaptar esse modelo pré-treinado para uma tarefa de geração de texto como sumarização ou compreensão de leitura de máquina seria apresentado para cada tarefa um dataset com pares de entradas/saídas desejadas: Z = $\{(x_i, y_i)\}$ i=1,..,N onde ambos x_i e x_i são sequências de tokens.

Durante um ajuste fino completo o modelo é inicializado com os pesos pré treinados ϕ_0 e atualizado para $\phi_0 + \Delta\phi$ repetidamente seguindo o gradiente para maximizar o objetivo de modelagem de linguagem condicional:

$$\max_{\Phi} \sum_{(x,y)\in\mathcal{Z}} \sum_{t=1}^{|y|} \log\left(P_\Phi(y_t|x, y_{<t})\right)$$

Um dos principais problemas de fazer o ajuste fino completo é que para cada tarefa específica é necessário aprender diferentes conjuntos de parâmetros $\Delta\phi$ cuja dimensão $|\Delta\phi|$ é iqual ϕ_0. Portanto, se o modelo pré-treinado é grande como o GPT-3, que possui cerca de 175 bilhões de parâmetros, armazenar e fazer o deploy de muitas instâncias de modelos ajustados pode ser inviável ou um desafio considerável.

O LoRA adota uma abordagem mais eficiente na parametrização, onde os parâmetros adicionais de uma tarefa específica são colocados em uma camada muito menor que o modelo original. Considerando o conjunto de parâmetros dessa camada

menor Θ. A otimização é feita sobre o conjunto de parâmetros Θ, mantendo o restante dos parâmetros congelados.

$$\max_{\Theta} \sum_{(x,y)\in\mathcal{Z}} \sum_{t=1}^{|y|} \log\left(p_{\Phi_0+\Delta\Phi(\Theta)}(y_t|x, y_{<t})\right)$$

2.17.2 Funcionamento do LoRA na atualização dos pesos

Embora o trabalho [19] tenha focado em fazer experimentos em pontos específicos de modelos que usam a arquitetura *Transformer*, o LoRA pode ser utilizado em qualquer camada densa nos modelos de aprendizado profundo.

Um modelo pode conter muitas camadas densas que fazem multiplicação de matrizes, essas matrizes de pesos normalmente são *full-rank* quando são adaptadas para tarefas especificas, o trabalho [1] mostra que modelos de linguagem pré-treinados tem uma baixa dimensão intrínsica e pode aprender de forma eficiente mesmo utilizando aplicando a atualização de pesos em uma pequena quantidade de parâmetros. Para um modelo pré-treinado com matriz de peso $W_0 \in \mathbb{R}^{d\times k}$, a atualização de pesos é restrita a uma decomposição de matrizes de dimensões pequenas, que são a matrizes A e B na seguinte representação: $W_0 + \Delta W = W_0 + BA$, onde $B \in \mathbb{R}^{d\times r}$, $A \in \mathbb{R}^{r\times k}$, e $r \ll \min(d, k)$.

Durante o treinamento W_0 é congelado e não recebe atualizações de gradiente, enquanto A e B contém os parâmetros treináveis, ambos W_0 e $\Delta W = BA$ são multiplicados com a mesma entrada, então seus respectivos vetores de saídas são somados.

Quando colocado em produção é possível computar e armazenar $W = W_0 + BA$ e executar a inferência normalmente. Ambos W_0 e BA estão em $\mathbb{R}^{d\times k}$, então para para fazer a troca para outra tarefa é possível recuperar W_0, fazendo a subtração de BA e adicionando um $B'A'$ diferente, que é uma operação rápida com pouco consumo de memória. Isso garante que não seja introduzido latência adicional durante as inferências comparado ao modelo com ajuste fino completo.

Uma dimensão intrínseca baixa significa que os dados podem ser efetivamente representados ou aproximados por um espaço de dimensão inferior, mantendo a maior parte de sua informação ou estrutura essencial, em outras palavras, isso significa que podemos decompor a nova matriz de pesos para a tarefa adaptada em matrizes de dimensões inferiores sem perder muitas informações importantes.

Por exemplo, suponha que ΔW seja a atualização de peso para uma matriz de peso $A \times B$. Então, podemos decompor a matriz de atualização de peso em duas matrizes menores: $\Delta W = W_A W_B$, onde W_A é uma matriz $A \times r$-dimensional e W_B é uma matriz $r \times B$-dimensional. Aqui, mantemos o peso original W congelado e treinamos apenas as novas matrizes W_A e W_B . Em resumo, este é o método LoRA, ilustrado na figura abaixo.

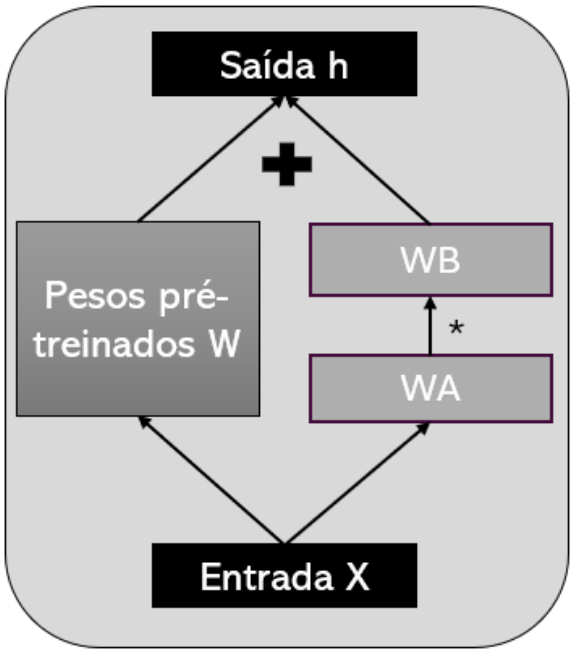

Figura 2.11: Esquema de atualização das matrizes com o LoRA

Algumas das características mais relevantes do LoRA é a redução da memória RAM, tempo de processamento e espaço de armazenamento. Para modelos grandes baseado no *Transformer*, o LoRA reduz o uso de VRAM em até 2/3 do total pois não é necessário armazenar no otimizador os estados dos parâmetros congelados. No GPT-3 175B, foi reduzido o uso de VRAM durante o treinamento de 1.2TB para 350GB.

Ainda no GPT-3 com o $r = 4$ e somente as matrizes de *Query* e *Value* sendo adaptadas, o tamanho do checkpoint reduziu de 350GB para 35MB. Isso permite treinar com muito menos GPUs e evitar gargalos de I/O. Outro benefício é que é possível alterar entre tarefas com muito menos custo, visto que só é necessário fazer o deploy somente dos pesos LoRA e não de todos os pesos do modelo original. Isso permite a criação de muitos módulos LoRA customizados que podem ser trocados rapidamente em máquinas que armazenam os pesos pré-treinados em VRAM.

Foi observado um aumento de 25% da velocidade do treino no GPT-3 175B comparado ao ajuste fino completo, visto que não é necessário calcular o gradiente para a maior parte dos parâmetros. Outras vantagens do LoRA são:

- Um modelo grande pré-treinado pode ser compartilhado ao construir vários pequenos módulos LoRA para diferentes tarefas específicas. É possível congelar o modelo compartilhado e de forma eficiente trocar os módulos LoRA treinados para tarefas específicas. O que reduz o custo de armazenamento e o custo para processar tarefas diferentes.
- Torna o treinamento mais eficiente reduzindo a necessidade de hardware em até 3 vezes, pois não é necessário calcular o gradiente e atualizar os pesos da maior parte

dos parâmetros. O que é necessário é apenas otimizar uma camada muito menor que foi injetada.

- Permite fazer o merge entre as matrizes treinadas com os pesos congelados ao fazer o deploy, com isso não há aumento do tamanho do modelo e da latência da inferência.
- O LoRA pode ser utilizado em conjunto com outros métodos existentes.

Caso seja feito o merge das matrizes A.B em W para eliminar a latência adicional se perde a versatilidade de compartilhar um único modelo original com diferentes módulos LoRA. E caso se opte em não fazer o merge para se aproveitar do dinamismo em reaproveitar o modelo original terá uma latência extra na inferência embora pequena.

Experimentos realizados

Os autores do LoRA realizaram alguns experimentos para entender quais matrizes da arquitetura *Transformer* vale mais a pena focar durante o ajuste fino, e também para verificar qual seria o tamanho ótimo das matrizes de baixa dimensão que irão ter seus parâmetros atualizados.

No primeiro experimento foi definido um limite máximo de parâmetros que poderiam ser atualizados com o LoRA. A expectativa ao realizar o experimento é descobrir dado um limite de parâmetros a serem atualizados quais matrizes ou combinações de matrizes da arquitetura *Transformer* trará os melhores resultados [19]. Nesse experimento foram considerados apenas as matrizes que envolvem o módulo *self-attention* da arquitetura *Transformer* e a quantidade de parâmetros definida foi 18 milhões, que quando armazenados em FP16 tem em torno de 35MB [19].

A tabela 2.4 foi retirada o paper original do LoRA [19].

Tabela 2.4: Resultados dos experimentos com diferentes combinações de matrizes e r

	# of Trainable Parameters = 18M						
Weight Type	W_q	W_k	W_v	W_o	W_q, W_k	W_q, W_v	W_q, W_k, W_v, W_o
Rank r	8	8	8	8	4	4	2
WikiSQL ($\pm 0.5\%$)	70.4	70.0	73.0	73.2	71.4	**73.7**	**73.7**
MultiNLI ($\pm 0.1\%$)	91.0	90.8	91.0	91.3	91.3	91.3	**91.7**

Os ajustes finos foram feitos nos conjuntos de dados WikiSQL e MultiNLI e no modelo GPT-3 175B. Os autores do LoRa concluíram que adaptar as matrizes Wq e Wv com *r*=4 trouxe os melhores resultados no geral. É notório que colocar todos os parâmetros adaptando Wq ou Wk resulta em um desempenho significativamente inferior Isto sugere que mesmo um *r*=4 possa capturar informações suficientes em uma matriz é

preferível adaptar mais matrizes de peso com um *r* menor do que adaptar um única matriz com um *r* maior [19].

O segundo experimento tinha como objetivo é descobrir o quanto que aumentar o *r* aumenta a performance ou acerto do modelo na inferência, mas nesse caso sem aplicar nenhum limite de quantidade máxima de parâmetros a serem atualizados. Os ajustes finos foram feitos nos conjuntos de dados WikiSQL e MultiNLI e no modelo GPT-3 175B. A tabela 2.5 foi retirada o paper original do LoRA [19].

Tabela 2.5: Resultados dos experimentos com diferentes combinações de matrizes e r

	Weight Type	$r=1$	$r=2$	$r=4$	$r=8$	$r=64$
WikiSQL($\pm$0.5%)	W_q	68.8	69.6	70.5	70.4	70.0
	W_q, W_v	73.4	73.3	73.7	73.8	73.5
	W_q, W_k, W_v, W_o	74.1	73.7	74.0	74.0	73.9
MultiNLI ($\pm$0.1%)	W_q	90.7	90.9	91.1	90.7	90.7
	W_q, W_v	91.3	91.4	91.3	91.6	91.4
	W_q, W_k, W_v, W_o	91.2	91.7	91.7	91.5	91.4

Foi concluído que de forma geral as matrizes realmente possuem uma dimensão intrínseca baixa, tendo o LoRA um ótimo desempenho com um *r* muito pequeno, com destaque a combinação Wq, Wv e *r*=2, onde aumentar o *r* não trouxe ganhos significativos [19].

2.18 AdaLora (*Adaptive Low-Rank Adaptation*)

Várias técnicas foram propostas para se fazer atualizações incrementais nos pesos pré-treinados de forma mais eficiente. Esses métodos normalmente distribuem através das matrizes de pesos o conjunto limitado de parâmetros a serem treinados, mas fazem isso ignorando a diferente importância que cada parâmetro ou matriz de peso tem, como resultado o ajuste fino se torna sub-ótimo [66].

A pergunta que os autores do AdaLoRA tinha inicialmente era: Como alocar o orçamento dos parâmetros (que é número total de treináveis parâmetros) de forma adaptativa de acordo com a importância dos módulos para melhorar o desempenho do ajuste fino com eficiência de parâmetros?

Para responder a esta questão, foi proposto o AdaLoRA, que aloca dinamicamente o orçamento dos parâmetros entre matrizes de peso durante o ajuste fino. Especificamente, AdaLoRA ajusta o *r* das matrizes adicionadas para controlar o orçamento. Para matrizes importantes são atribuídas com *r* mais alto para que possam capturar informa-

ções mais refinadas e específicas da tarefa. Os de menor importância passam a ter um r mais baixo para economizar o orçamento computacional [66].

Foram conduzidos extensos experimentos com vários modelos pré-treinados em processamento de linguagem natural, nas tarefas de perguntas e respostas e geração de linguagem natural para validar a eficácia do AdaLoRA. Os resultados demonstram que AdaLoRA manifesta melhorias notáveis, especialmente em ambientes de baixo orçamento.

Os problemas citados na distribuição dos parâmetros de atualização sem considerar a importância final das matrizes de pesos é uma das limitações do LoRA, onde o r é igual para todas as matrizes de pesos. A figura 2.12 abaixo compara a performance do LoRA com diferentes configurações de ajuste fino através dos módulos da arquitetura *Transformer*, os testes atualizam sempre a mesma quantidade de parâmetros.

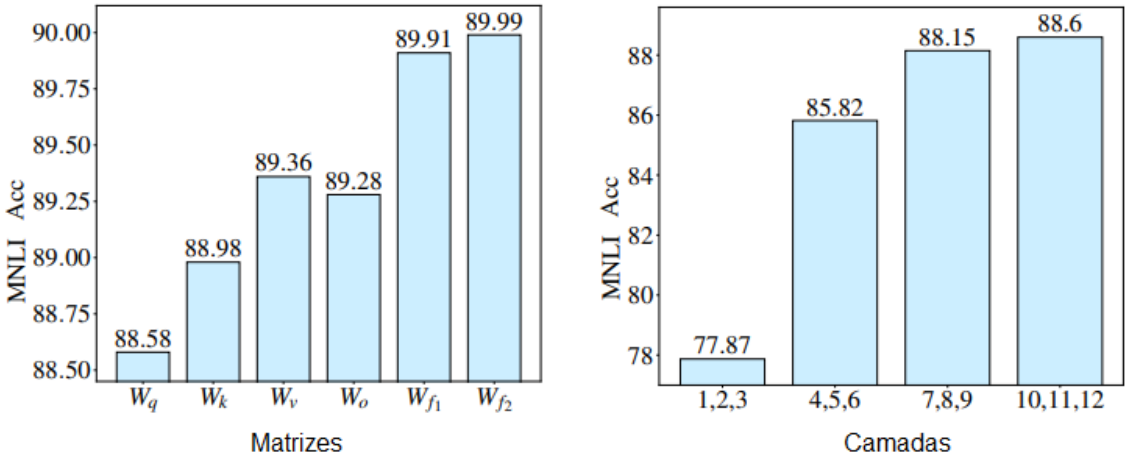

Figura 2.12: Acurácia do LoRA em diferentes configurações, fonte: [66].

O gráfico da esquerda da (2.12b) mostra que as camadas *FFN* no ajuste fino alcançam melhor desempenho do que os módulos de *self-attention*. Além disso, o gráfico da direita(2.12a) demonstra que as matrizes de peso nas camadas superiores são mais importantes do que as das camadas inferiores. E os resultados de distribuição que os testes com o AdaLoRA foram coerentes com os experimentos feitos diretamente com o LoRA.

Adicionar mais parâmetros treináveis às matrizes de peso relevantes pode levar a um melhor desempenho do modelo. Por outro lado, adicionar mais parâmetros a essas matrizes de peso menos importantes produz ganhos marginais ou até prejudica o desempenho do modelo. Dado o orçamento de parâmetros, é preferível alocar mais parâmetros para esses módulos importantes.

Distribuir o orçamento uniformemente para todas as matrizes/camadas de peso, como LoRA e outros métodos geralmente proporciona um desempenho abaixo do ideal.

2.18.1 Experimentos

Foram realizados uma série de testes para validar a eficácia do AdaLoRA com comparativos com o LoRA e outras técnicas existentes, os testes de ajuste fino foram realizados nos modelos DeBERTaV3-base e BART-large, e foi utilizado um conjunto de dados que avalia a compreensão da linguagem natural (GLUE), perguntas e respostas (SQuADv1 e SQuADv2) e também geração de linguagem natural (XSum e CNN/DailyMail)

DeBERTaV3-base + GLUE

A tabela 2.6 foi retirada o paper original do AdaLoRA [66].

Tabela 2.6: Testes com o modelo DeBERTaV3-base + conjunto de dados GLUE

Method	**# Params**	**MNLI** m/mm	**SST-2** Acc	**CoLA** Mcc	**QQP** Acc/F1	**QNLI** Acc	**RTE** Acc	**MRPC** Acc	**STS-B** Corr	**All** Ave.
Full FT	184M	89.90/90.12	95.63	69.19	**92.40/89.80**	94.03	83.75	89.46	91.60	88.09
BitFit	0.1M	89.37/89.91	94.84	66.96	88.41/84.95	92.24	78.70	87.75	91.35	86.02
HAdapter	1.22M	90.13/90.17	95.53	68.64	91.91/89.27	94.11	84.48	89.95	91.48	88.12
PAdapter	1.18M	90.33/90.39	95.61	68.77	92.04/89.40	94.29	85.20	89.46	91.54	88.24
LoRA$_{r=8}$	1.33M	90.65/90.69	94.95	69.82	91.99/89.38	93.87	85.20	89.95	91.60	88.34
AdaLoRA	1.27M	**90.76/90.79**	**96.10**	**71.45**	**92.23/89.74**	**94.55**	**88.09**	**90.69**	**91.84**	**89.31**
HAdapter	0.61M	90.12/90.23	95.30	67.87	91.65/88.95	93.76	85.56	89.22	91.30	87.93
PAdapter	0.60M	90.15/90.28	95.53	69.48	91.62/88.86	93.98	84.12	89.22	91.52	88.04
HAdapter	0.31M	90.10/90.02	95.41	67.65	91.54/88.81	93.52	83.39	89.25	91.31	87.60
PAdapter	0.30M	89.89/90.06	94.72	69.06	91.40/88.62	93.87	84.48	89.71	91.38	87.90
LoRA$_{r=2}$	0.33M	90.30/90.38	94.95	68.71	91.61/88.91	94.03	85.56	89.71	**91.68**	88.15
AdaLoRA	0.32M	**90.66/90.70**	**95.80**	**70.04**	**91.78/89.16**	**94.49**	**87.36**	**90.44**	91.63	**88.86**

Os melhores resultados em cada conjunto de dados é mostrado em negrito. No gráfico "Full FT", "HAdapter"and "PAdapter"representam ajuste fino completo, Houlsby Adapter e Pfeiffer Adapter respectivamente. Os dados mostrados são a média de cinco execuções diferentes com a inicialização do pesos utilizando cinco seeds diferentes. A tabela 2.7 foi retirada o paper original do AdaLoRA [66].

Tabela 2.7: Testes com o modelo DeBERTaV3-base + conjunto de dados SQuAD

	SQuADv1.1				**SQuADv2.0**			
Full FT	86.0 / 92.7				85.4 / 88.4			
# Params	0.08%	0.16%	0.32%	0.65%	0.08%	0.16%	0.32%	0.65%
HAdapter	84.4/91.5	85.3/92.1	86.1/92.7	86.7/92.9	83.4/86.6	84.3/87.3	84.9/87.9	85.4/88.3
PAdapter	84.4/91.7	85.9/92.5	86.2/92.8	86.6/93.0	84.2/87.2	84.5/87.6	84.9/87.8	84.5/87.5
LoRA	86.4/92.8	86.6/92.9	86.7/93.1	86.7/93.1	84.7/87.5	83.6/86.7	84.5/87.4	85.0/88.0
AdaLoRA	**87.2/93.4**	**87.5/93.6**	**87.5/93.7**	**87.6/93.7**	**85.6/88.7**	**85.7/88.8**	**85.5/88.6**	**86.0/88.9**

A figura mostra os resultados do ajuste fino do DeBERTaV3-base nos conjunto de dados SQuAD v1.1 e SQuADv2.0. A linha "Params"significa a porcentagem de parâmetros treináveis em relação ao ajuste fino completo, os resultados foram reportados em EM (Exact match) e F1 e os melhores resultados estão em negrito.

Nesses testes o AdaLoRA supera consistentemente as abordagens comparadas em todos as variações de quantidade de parâmetros treináveis nas duas métricas utilizadas EM e F1.

BART-large + XSum e CNN/DailyMail

Para fazer um comparativo com o estado da arte em tarefas de geração de linguagem natural (NLG) foi feito o ajuste fino usando do AdaLoRA no modelo BART-large com dois conjuntos de dados XSum e CNN/DailyMail.

A coluna "Params"significa a porcentagem de parâmetros treináveis em relação ao ajuste fino completo. Os resultados foram reportados em ROUGE 1/2/L, e os melhores resultados estão em negrito. A tabela 2.8 foi retirada o paper original do AdaLoRA [66].

Tabela 2.8: Testes com o modelo BART-large + conjuntos de dados XSum e CNN/DailyMail

# Params	Method	XSum	CNN/DailyMail
100%	Full FT	**45.49 / 22.33 / 37.26**	44.16 / 21.28 / 40.90
2.20%	LoRA	43.95 / 20.72 / 35.68	**45.03** / 21.84 / 42.15
	AdaLoRA	**44.72 / 21.46 / 36.46**	45.00 / **21.89 / 42.16**
1.10%	LoRA	43.40 / 20.20 / 35.20	44.72 / 21.58 / 41.84
	AdaLoRA	**44.35 / 21.13 / 36.13**	**44.96 / 21.77 / 42.09**
0.26%	LoRA	43.18 / 19.89 / 34.92	43.95 / 20.91 / 40.98
	AdaLoRA	**43.55 / 20.17 / 35.20**	**44.39 / 21.28 / 41.50**
0.13%	LoRA	42.81 / 19.68 / 34.73	43.68 / 20.63 / 40.71
	AdaLoRA	**43.29 / 19.95 / 35.04**	**43.94 / 20.83 / 40.96**

Foi comparado o desempenho do ajuste fino em quatro níveis orçamentários de parâmetros em relação ao número total de parâmetros: 0,13%, 0,26% , 1,10% e 2,20%. O AdaLoRA alcança desempenho melhor ou igual em comparação com a linha de base em ambos os conjuntos de dados (XSum e CNN/DailyMail) em todos os níveis de orçamento.

2.19 IA3 (*Infused Adapter by Inhibiting and Amplifying Inner Activations*)

O trabalho [30] tinha como objetivo comparar *few-shot* em aprendizagem em contexto (*ICL*) [34] com técnicas de ajuste fino com eficiência de parâmetros e para isso criaram a técnica chamada IA3, que resumidamente dimensiona ativações com vetores aprendidos, com isso obtém um bom desempenho introduzindo uma quantidade relativamente pequena de novos parâmetros no modelo.

De acordo com os autores, dado que técnicas de ajuste fino com eficiência de parâmetros permite que um modelo seja adaptado a uma nova tarefa com requisitos de armazenamento e custo computacional relativamente pequenos, então essas técnicas poderiam ser uma alternativa promissora ao *ICL*.

Para comparar de forma correta com *few-shot ICL* é necessário uma técnica de ajuste fino que tenha as seguintes propriedades:

- Deve adicionar ou atualizar o menor número possível de parâmetros para evitar custos de armazenamento e memória.

- Deve atingir grande precisão após treinamento com poucos exemplos rotulados em novas tarefas.
- Permitir lotes de tarefas mistas, uma vez que esse é um recurso do ICL. Para permitir facilmente lotes de tarefas mistas, a técnica idealmente não deveria modificar o modelo em si, caso contrário, cada exemplo em um lote precisaria efetivamente ser processado por um modelo ou gráfico computacional diferente. Uma alternativa mais conveniente é fornecida por métodos que modificam diretamente as ativações do modelo, uma vez que isso pode ser feito de forma independente e barata para cada exemplo do lote de acordo com a tarefa a que o exemplo corresponde.

O trabalho [30] não tinha como objetivo apenas criar a técnica IA3, mas sim propor uma receita composta por um modelo, uma técnica de ajuste fino com eficiência de parâmetros e um conjunto fixo de hiper-parâmetros que tenha forte desempenho em tarefas novas com poucos exemplos rotulados, ao mesmo tempo que permita lotes de tarefas mistas durante a inferência o que incorre em custos computacionais e de armazenamento mínimos. E que essa receita tivesse resultados comparáveis ao ajuste fino completo enquanto atualiza apenas uma pequena fração dos parâmetros do modelo.

Idealmente, o modelo pré-treinado a ser escolhido deveria atingir alto desempenho em novas tarefas após ajuste fino em um número limitado de exemplos rotulados. Em experimentos preliminares aplicando métodos de ajuste fino com eficiência a diferentes modelos pré-treinados, obtiveram o melhor desempenho com T0 [49]. A abordagem proposta, ou a receita citada anteriormente, se baseou no modelo T0 [49], uma variante do T5 [45] que recebeu um ajuste fino em multitarefa a partir de conjunto de dados contendo prompts. Embora T0 tenha sido projetado para generalização zero-shot, foi demonstrado que ele também atinge um forte desempenho após o ajuste fino com apenas alguns exemplos rotulados.

2.19.1 Detalhes do IA3

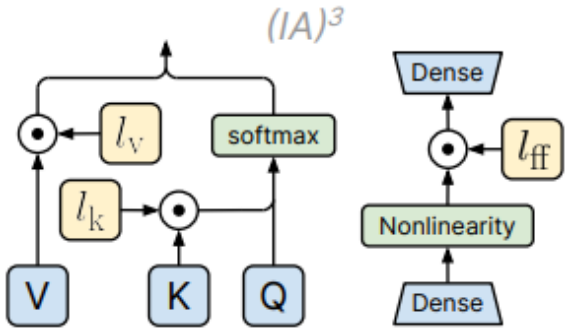

Figura 2.13: Estrutura básica do IA3 fonte:[30]

IA3 introduz o vetores aprendidos l_k, l_v e l_{ff} que respectivamente redimensionam as matrizes *key* e *value*, por meio da multiplicação elemento a elemento, nos mecanismos de atenção e as ativações internas em redes *feed-forward*.

O treinamento na técnica IA3 acrescenta alguns aspectos com o intuito de melhorar a performance:

Treinamento de improbabilidade (*Unlikelihood Training*)

Foram explorados dois termos de perda (*loss terms*) adicionais para melhorar o desempenho do ajuste fino de modelos de linguagem. Os modelos de linguagem são normalmente treinados com perda de entropia cruzada L_{LM} onde o modelo é treinado para aumentar a probabilidade da sequência alvo correta y dada a sequência de entrada x. Para avaliação, foi usada a probabilidade que o modelo atribui à escolha correta quanto das probabilidades atribuídas pelo modelo às escolhas incorretas. Para considerar isso durante o treinamento, foi adicionado uma perda de probabilidade (*Unlikelihood loss*) [61] o que desencoraja o modelo de prever tokens de sequências alvo incorretas. A hipótese é que a adição de perda de probabilidade L_{UL} melhorará os resultados classificação porque o modelo será treinado para atribuir probabilidades mais baixas a escolhas incorretas, aumentando assim a chance de que a escolha correta seja classificada em primeiro lugar.

Normalização de comprimento (*Length Normalization*)

As possíveis sequências alvo para um determinado exemplo de treinamento podem ter comprimentos significativamente diferentes, a classificação pode favorecer escolhas mais curtas porque a probabilidade atribuída pelo modelo a cada token é ≤ 1 e quanto mais tokens maior é a tendência do valor final ficar menor. Para melhorar isso, foi considerada o uso da normalização de comprimento (L_{LN}), que divide a pontuação de cada opção de resposta possível pelo número de tokens da opção.

Ao treinar um modelo com L_{LM}, L_{UL} e L_{LN}, simplesmente é feita a soma dos mesmos, isso evita a introdução de quaisquer hiper parâmetros que seriam problemáticos para ajustar na configuração de few-shot (onde conjuntos de validação de tamanho realista são minúsculos por necessidade [40]).

Concluíram que adicionar L_{LN} melhora a precisão de 60,7% para 62,71% e incluir L_{UL} e L_{LN} proporciona uma melhoria adicional para 63,3%.

Resultados no conjuntos de dados RAFT (Real-world Few-shot Tasks)

Esse conjunto de dados foi criado para avaliar de forma explicita a performance do modelo ou técnica em aprendizado "few-shot". O RAFT é composto por conjunto de dados que possui apenas 50 exemplos de treino e um conjunto grande de dados para teste.

A tabela 2.9 foi retirada o paper original do IA3 [30] e possui a comparação da técnica IA3 com vários outras técnicas em vários conjuntos de dados.

Tabela 2.9: Comparação da técnica IA3 com vários outras técnicas em vários datasets.

	# of Param	COPA	H-Swag	StoryCloze	Winogrande	WSC	WiC	RTE	CB	ANLI-R1	ANLI-R2	ANLI-R3
Full Model Fine-tuning	3B	$81.0_{11.0}$	$46.4_{8.8}$	$93.8_{2.7}$	$56.5_{1.5}$	$65.4_{7.7}$	$57.7_{3.9}$	$79.8_{3.6}$	$87.5_{5.4}$	$46.6_{2.5}$	$41.3_{0.9}$	$40.2_{5.3}$
BitFit (with LayerNorm)	1.3M	$75.0_{2.0}$	$29.5_{3.6}$	$88.6_{0.7}$	$49.6_{1.3}$	$61.5_{11.5}$	$51.7_{2.2}$	$72.2_{1.1}$	$57.1_{1.8}$	$36.5_{0.8}$	$35.3_{2.2}$	$36.6_{0.8}$
LayerNorm	250K	$76.0_{2.0}$	$29.6_{3.4}$	$88.7_{0.9}$	$49.4_{1.4}$	$63.5_{12.5}$	$52.2_{1.6}$	$71.8_{0.4}$	$57.1_{1.8}$	$36.5_{0.7}$	$35.1_{2.6}$	$36.3_{1.0}$
Adapter	12.9M	$84.0_{3.0}$	$41.9_{3.8}$	$91.7_{3.7}$	$54.7_{3.6}$	$65.4_{1.0}$	$55.5_{2.7}$	$76.2_{3.6}$	$87.5_{3.6}$	$45.1_{2.6}$	$40.4_{1.2}$	$35.3_{1.3}$
Compacter	807K	$84.0_{5.0}$	$46.4_{2.5}$	$93.5_{2.2}$	$55.5_{2.9}$	$64.4_{6.7}$	$55.2_{3.8}$	$75.8_{6.1}$	$82.1_{3.6}$	$40.8_{3.3}$	$37.4_{0.2}$	$35.8_{3.3}$
Compacter++	540K	$86.0_{3.0}$	$46.3_{3.0}$	$93.5_{1.2}$	$55.1_{1.1}$	$65.4_{3.9}$	$54.1_{2.2}$	$76.9_{0.4}$	$82.1_{3.6}$	$41.7_{0.4}$	$38.3_{1.8}$	$36.9_{1.5}$
Prompt tuning (10)	41K	$67.0_{5.0}$	$29.9_{0.6}$	$84.2_{0.8}$	$51.9_{1.6}$	$54.8_{10.6}$	$51.6_{2.0}$	$52.7_{5.4}$	$66.1_{1.8}$	$34.2_{1.9}$	$33.5_{1.1}$	$33.5_{1.3}$
Prompt tuning (100)	409K	$60.0_{19.0}$	$26.8_{0.6}$	$74.0_{3.4}$	$51.1_{0.8}$	$60.6_{4.8}$	$50.0_{1.1}$	$48.0_{2.9}$	$53.6_{17.9}$	$33.4_{1.2}$	$33.8_{0.5}$	$33.3_{0.8}$
Prefix tuning	576K	$71.0_{8.0}$	$42.1_{4.0}$	$90.2_{3.1}$	$52.0_{1.3}$	$56.7_{3.3}$	$54.2_{3.3}$	$68.6_{3.3}$	$84.0_{1.8}$	$43.3_{4.1}$	$37.5_{1.2}$	$36.5_{1.5}$
FishMask (0.2%)	6M	$82.0_{5.0}$	$44.1_{4.2}$	$94.2_{1.8}$	$54.5_{2.1}$	$63.5_{4.8}$	$52.5_{3.3}$	$76.9_{4.7}$	$83.9_{3.6}$	$43.7_{0.3}$	$39.7_{1.4}$	$37.2_{1.1}$
FishMask (0.02%)	600K	$84.0_{6.0}$	$38.2_{3.6}$	$93.6_{0.7}$	$53.9_{2.8}$	$61.5_{1.0}$	$53.5_{1.3}$	$75.5_{5.4}$	$76.8_{3.6}$	$39.9_{0.9}$	$38.1_{2.0}$	$36.2_{1.8}$
Intrinsic SAID	500K	$77.0_{4.0}$	$36.7_{4.5}$	$89.3_{2.3}$	$52.7_{2.1}$	$61.5_{8.7}$	$55.0_{2.7}$	$69.0_{7.6}$	$80.4_{0.0}$	$40.4_{3.3}$	$35.4_{4.1}$	$35.5_{1.6}$
Intrinsic SAID	20K	$76.0_{4.0}$	$38.3_{6.4}$	$89.7_{2.7}$	$50.9_{1.0}$	$55.8_{6.7}$	$55.3_{0.5}$	$66.1_{5.4}$	$83.9_{1.8}$	$41.3_{1.3}$	$38.5_{1.8}$	$35.8_{2.0}$
LoRA	9.1M	$88.0_{5.0}$	$47.1_{3.2}$	$93.6_{2.1}$	$56.8_{3.3}$	$60.6_{5.8}$	$55.2_{5.0}$	$78.3_{7.6}$	$85.7_{1.8}$	$45.1_{2.5}$	$41.0_{1.4}$	$39.5_{4.8}$
$(IA)^3$	540K	$87.0_{3.0}$	$49.4_{4.6}$	$94.7_{2.7}$	$59.8_{0.6}$	$68.3_{6.7}$	$56.0_{4.6}$	$78.0_{2.5}$	$87.5_{1.8}$	$48.6_{2.0}$	$40.8_{1.5}$	$40.8_{2.3}$

CAPÍTULO 3

A Abordagem proposta neste trabalho para os Experimentos

Este Capítulo descreve a estruturação metodológica dos experimentos realizados neste trabalho, tornando o processo replicável e os resultados passíveis de serem analisados por outros. As próximas seções detalham esse processo.

3.1 Técnicas e recursos a serem utilizados nos experimentos

Visando testar e avaliar a aplicação de técnicas de ajuste fino em modelos de linguagem, foram realizados vários experimentos de ajuste fino. Esses experimentos envolveram três modelos de linguagem de tamanhos variados, quatro diferentes técnicas de ajustes finos e, finalmente, foram utilizados três conjuntos de dados de classificação de textos com características diferentes. O trabalho realizou um total de trinta e seis experimentos a fim de satisfazer todas as combinações das triplas (LLM, Técnica de ajuste fino e conjunto de dados).

As técnicas de ajuste fino com eficiência de parâmetros utilizadas são o LoRA [19], AdaLoRA [66] e IA3 [30], além do ajuste fino completo tradicional, totalizando quatro métodos diferentes de ajuste fino.

Os modelos de linguagem utilizados são o DistilBERT [48] com sessenta e seis milhões de parâmetros, o RoBERTa [31] com cento e vinte cinco milhões e TinyLlama [65] com 1.1 bilhões.

Para cada combinação de técnicas de ajuste fino e de modelo de linguagem e são realizados testes com três conjunto de dados: o AG News [67], o IMDb [11] e o SNLI [5]. Todos esses conjuntos de dados estão relacionados à classificação de texto, mas com algumas diferenças em suas características. O IMDb [11] classifica resenhas de filmes em duas classes sendo elas positivo e negativo. O AG News [67] classifica notícias em quatro classes diferentes sendo elas mundo, esportes, negócios e ciência/tecnologia. O SLNI [5] é

um conjunto de dados que classifica a relação entre duas sentenças em implicação, neutro ou contradição.

3.2 Aspectos a serem avaliados nos experimentos

Um aspecto fundamental a ser observado nos experimentos é a análise da acurácia que cada uma das técnicas de ajuste fino apresenta. Este aspecto é o critério principal, pois sem acurácia no resultado, de nada adianta o sucesso em outros aspectos.

Técnicas que demandam menor poder computacional devem ser classificadas como promissoras. Assim, a exigência de poder de processamento é um aspecto importante a ser avaliado.

Alguns trabalhos correlatos, como o de Lialin e outros [29], concluíram que as técnicas de ajuste fino com eficiência de parâmetros possuem resultados variados a depender do tamanho do modelo. Lianlin e outros observaram uma forte tendência das técnicas de ajuste fino serem mais eficientes e vantajosas em modelos maiores [29]. Dado que as LLM maiores podem oferecer maior gama de funcionalidades, esse critério de avaliação é também importante de ser observado neste trabalho.

3.3 Medidas a serem obtidas dos experimentos

Durante cada ajuste fino serão monitorados:

A) O consumo de memória da GPU. O monitoramento do uso de memória da GPU será realizada pela nvidia-ml-py3 [8] que permite via código Python verificar o uso da memória a qualquer momento. Durante o treinamento a cada trinta segundos é verificado o uso de memória e, ao final do treinamento, é feito a média aritmética simples para obtenção do resultado;
B) O tempo total de treinamento;
C) Ao final dos testes os modelos, serão avaliados no conjunto de dados de teste a acurácia resultante do ajuste fino realizado. A acurácia, basicamente, verifica a quantidade total de classificações corretas e erradas feitas pelo modelo. O cenário ideal é o modelo acertar todas as classificações alcançando com isso uma acurácia de 100%. A acurácia foi monitorada a cada época tanto no conjunto de dados de teste quanto no de treino para verificação de possível *overfitting*.

Os resultados computados serão agrupados por modelo e conjunto de dados, onde, para cada um dos elementos desse agrupamento, é comparado os resultados obtidos utilizando as técnicas de ajuste fino LoRA, AdaLoRA, IA3 e ajuste fino.

O objetivo nos experimentos é gerar uma massa de resultados que permita comparar o uso de memória, tempo de treinamento e a acurácia de cada uma das técnicas.

A avaliação dos resultados envolve uma certa complexidade, visto que todas as três características avaliadas são importantes, embora algumas mais e outras menos importantes, a depender do cenário. Por exemplo: se uma técnica que oferece alguma vantagem sobre as outras no uso de memória RAM, sacrificando, mesmo que pouco, o tempo de processamento e a acurácia, ela pode ser a melhor ou única opção caso a GPU disponível não tenha memória o suficiente. Por outro lado, esta técnica poderá ser a pior opção, caso a acurácia e tempo de processamento sejam a prioridade.

Da mesma forma que o cenário citado no exemplo anterior, facilmente pode-se criar inúmeras situações onde diferentes cenários gerarão diferentes critérios de avaliação do experimento.

Outro aspecto dificultador da análise é o conjunto relativamente extenso de resultado, o qual terá resultados e trinta e seis experimentos. Mas, por outro lado, essa abundância de experimentos permite uma avaliação detalhada de possibilidades de combinação dos recursos utilizados, sendo esta a maior contribuição deste trabalho.

3.4 Recursos computacionais para realização dos experimentos

Todos os testes foram realizados em apenas uma GPU A100 na máquina DGX-A100 disponibilizada pela UFG. Essa máquina possui as seguintes especificações:

- 8 x GPU NVIDIA A100 80GB.
- 1 TB de memória RAM.
- 2 x 64-Core AMD Rome 7742

Para a execução dos testes, cada experimento teve sua implementação escrita na forma de um código em *Python*. Foram trinta e seis diferentes configurações para executar cada uma das combinações de ajuste fino.

As execuções na máquina DGX foram feitas através de um *container docker* que foi montado com as seguintes configurações: O *container* foi inicializado utilizando uma imagem docker disponibilizada pela NVIDIA: nvcr.io/nvidia/ai-workbench/python-cuda122:1.0.3, que possui como sistema operacional o Ubuntu 22.04 com o python instalado na versão 3.10 e CUDA 12.2.

O código Python escrito usa o *framework hugginface*, que suporta nativamente os modelos selecionados e as técnicas de ajuste fino. Esse *framework* possui algumas dependências listadas abaixo no formato nome / versão que serão instaladas no *container* gerado pela imagem base da NVIDIA:

- datasets / 2.18.0
- transformers / 4.36.2
- peft / 0.9.0
- evaluate / 0.4.1
- accelerate / 0.27.2
- scikit-learn / 1.4.1.post1
- nvidia-ml-py3 / 7.352.0

Os testes foram realizados de forma sequencial em uma única GPU da máquina utilizada.

Foram executadas três épocas completas para cada um dos ajustes finos, sendo o *batch size* igual a seis para todos os modelos, embora fosse possível aumentar o *batch* para os modelos DistilBERT [48] e RoBERTa [31] para 128 ou mais. Para modelo TinyLlama [65], na estrutura computacional utilizada, o *batch size* máximo possível é algo próximo de seis. Então, no intuito de se obter uma melhor comparação, todos os modelos foram executados com o *batch size* igual a seis.

Para todas as técnicas de ajuste fino com eficiência de parâmetros foram selecionados as matrizes *Query* e *Value* da arquitetura *transformer* que é a base dos 03 modelos testados, como alvo do ajuste fino.

Para o LoRA [19] e AdaLoRA [66], o rank selecionado foi $r = 4$. Vale ressaltar que o AdaLoRA pode redistribuir de forma adaptativa o r para cada uma das matrizes alvo a fim de melhorar o resultado final.

A escolha das matrizes *Query* e *Value* e do *rank* = *4* foi baseada nos resultados do próprio trabalho original do LoRA [19], onde foi demonstrado que um *rank* maior que 4 não trouxe resultados adicionais condizentes com o aumento da processamento. E, pelo menos, nos modelos testados no trabalho de Hu e outros, a melhor combinação de dois módulos *transformer* para o *rank=4* foram os módulos *Query* e *Value*.

O próximo capítulo apresenta e analisa os resultados obtidos com os experimentos realizados segundo os critérios e padrões definidos neste capítulo.

CAPÍTULO 4

Resultados e Análises

Conforme descrito no Capítulo anterior, os resultados serão avaliados por três óticas diferentes: consumo de memória da GPU, tempo de execução e a acurácia dos resultados. Sabe-se que dependendo do problema a ser tratado, é possível que diferentes combinações testadas podem ser mais convenientes. Assim, as seções a seguir apresenta os resultados e uma análise de cada um dos grandes critérios adotados.

Um aspecto importante de se ter em mente quando avaliar cada critério de análise, são:

- O tamanho dos modelos de linguagem, que, conforme descrito anteriormente, são: DistilBERT com 66M de parâmetros, o RoBERTa com com 125M de parâmetros e TinyLlama com 1.1B de parâmetros.
- O tamanho dos conjunto de dados utilizados, bem como a natureza deles é uma informação importante para a analise. Resumidamente, o IMDb possui cinquenta mil resenhas de filmes, o AG News contém trinta mil amostras de notícias coletadas da web e o SNLI possui quinhentos e setenta mil pares de frases em diferentes línguas.

4.1 Resultados e Análises do uso de memória da GPU

Os resultados obtidos com relação ao quesito uso de memória da GPU estão apresentados na Tabela 4.1 sabendo-se o tamanho de cada modelo de linguagem testado, percebe-se que a quantidade de parâmetros do modelo, naturalmente, impacta no uso de memória. Entretanto, essa quantidade pode variar com o método de ajuste fino e com o conjunto de dados utilizado.

Tabela 4.1: Resultados de uso de memória em gigabyte por experimento.

Ajuste fino	DistilBERT			RoBERTa			TinyLlama		
	IMDb	AG News	SNLI	IMDb	AG News	SNLI	IMDb	AG News	SNLI
LoRA	2,7	2,7	2,8	4,6	4,6	4,7	46,1	46,1	51,1
AdaLoRA	2,6	2,6	2,7	4,4	4,5	4,5	42,7	42,9	51,8
IA3	2,6	2,6	2,7	4,4	4,5	4,5	42,9	42,9	51,2
Completo	3,6	3,7	3,7	6,0	6,1	6,1	63,8	63,7	63,7

Pode-se observar na tabela que a magnitude do uso de memória para as técnicas de ajuste fino eficientes são muito próximas uma das outras. A mesma análise é possível de ser feita quando se considera o conjunto de dados utilizado, onde o uso de memória praticamente não varia com a mudança do conjunto de dados. Essas observações sugerem que o modelo de linguagem é o maior responsável por definir o uso de memória da GPU.

Outro aspecto importante que reforça a análise anterior é a manutenção de uma relação direta entre a quantidade de parâmetros com o uso da memória. Tomando o DistilBERT como referência temos a seguinte relação: DistilBERT terá 1,0 unidade de parâmetro e gasto de 1,0 unidade de memória, o RoBERTa terá 1,7 unidades de parâmetro e gasto de 1,89 unidades de memória e o TinyLlama terá 17,1 unidades de parâmetros e gasto de 16,6 unidades de memória.

Observa-se na tabela que o processo de um ajuste fino completo, realmente consome um montante de memória a mais, sendo cerca de 30% com o DistilBERT, 33% com o RoBERTa e, para o TinyLlama o consumo de memória é cerca de 38% para os conjuntos de dados IMDb e AG News, e de 25% para o SNLI.

Na combinação LoRA e TinyLlama com os conjuntos de dados IMDb e AG News houve um consumo de memória 10% maior do que nas demais técnicas de ajuste fino eficientes, um comportamento inesperado e que não foi observado nos outros modelos, e não foi possível identificar se essa exceção pode ter sido causada por um erro no gerenciamento de memória pelas ferramentas utilizadas.

Da análise da tabela 4.1 pode-se considerar que as técnicas de ajuste fino que se propõem a economizar memória, realmente conseguem reduzir o uso de memória, embora os testes realizados não mostrou a economia de memória descrita nos trabalhos que propuseram as técnicas, provavelmente a causa disso é o tamanho dos modelos, que nesse trabalho são menores.

Assim, dado uma estrutura computacional com restrições de memória, deve-se avaliar qual modelo de linguagem e técnica de ajuste fino é possível de ser utilizado. Obviamente, quanto maior o modelo de linguagem provavelmente maior acurácia será obtida.

4.2 Resultados e Análises do tempo de processamento dos experimentos

Os resultados obtidos com relação ao quesito tempo de processamento estão apresentados na Tabela 4.2 da mesma forma que na análise anterior, espera-se que os volumes de dados envolvidos no modelo de linguagem e nos conjuntos de dados sejam os responsáveis pelo tempo de processamento. A análise considera este aspecto, mas avaliará se as variações de tempo são proporcionais ao tamanho dos modelos e aos volumes de dados manipulados.

Tabela 4.2: Resultados de tempo dispendido, em minutos, para o processamento dos experimento.

Ajuste fino	DistilBERT			RoBERTa			TinyLlama		
	IMDb	AG News	SNLI	IMDb	AG News	SNLI	IMDb	AG News	SNLI
LoRA	9	36	164	16	63	289	541	2.086	7.938
AdaLoRA	10	42	193	19	76	340	544	2.114	9.030
IA3	9	34	154	15	59	267	534	2.058	8.536
Completo	10	40	181	17	71	320	583	2.283	7.824

Pode-se observar na tabela que o tempo de processamento, ao contrário do uso de memória da GPU, possui uma maior heterogeneidade nos resultados observados quando se compara as técnicas de ajuste fino eficientes com o ajuste fino completo. O tempo de processamento com o modelo de linguagem TinyLlama é acentuadamente maior que os demais modelos. O tempo total de treino do DistilBERT, RoBERTa e TinyLlama foram 12 horas, 22horas e 629horas respectivamente. Embora o TinyLlama seja em torno de nove vezes maior, com relação ao número de parâmetros, que o RoBERTa, o tempo total de treinamento foi quase 29 vezes maior. Mesmo estabelecendo uma proporção com a quantidade de parâmetros dos modelos, TinyLlama foi cerca de 3,1 vezes mais demorado que os demais. Percebe-se que o tempo de processamento não teve um comportamento linear com a quantidade de parâmetros do modelo, e sim um comportamento exponencial.

O comportamento da duração dos experimentos com o DistilBERT e o RoBERTa, foram praticamente idênticos se fizer uma proporção utilizando a quantidade de parâmetro de cada modelo linguagem e o tamanho dos conjuntos de dados utilizados. A correlação obtida com essa correção foi de 99,99% de semelhança nos tempos de execução medidos.

Assim, a análise do tempo de execução sugere que o tempo de processamento do experimento foi em função do tamanho modelo de linguagem e do conjunto de dados utilizado e também que redução no tempo de processamento não é um motivo adequado

para para a escolha das técnicas de ajuste fino eficientes, já que não demonstraram superioridade neste aspecto.

4.3 Resultados e Análises da acurácia obtida nos experimentos

Para realizar a análise da acurácia obtida nos experimentos, a Tabela 4.3 apresenta acurácia obtida nos dados de teste utilizados em cada experimento.

Vale ressaltar que a quantidade de classes nos conjuntos de dados são diferentes, embora todas sejam relacionadas a classificação de texto. Enquanto o IMDb classifica sentimentos em duas classes, o AG News faz classificação do tipo de notícia em quatro 4 classes e o SNLI que faz inferência lógica, tem três classes. Embora diferentes, todas as tarefas são realizadas com um número relativamente baixo de classes dado o poder de classificação das redes utilizadas.

Tabela 4.3: Resultados das acurácias, em %, obtidas nos experimentos.

-	DistilBERT			RoBERTa			TinyLlama			-
Ajuste fino	IMDb	AG News	SNLI	IMDb	AG News	SNLI	IMDb	AG News	SNLI	Média
LoRA	88,8	91,8	81,3	94,3	93,2	88,1	96,1	94,0	90,9	90,9
AdaLoRA	87,2	90,8	81,4	93,6	92,7	87,9	95,7	93,2	91,1	90,4
IA3	85,0	89,8	74,1	90,9	91,4	83,9	92,6	92,3	87,6	87,5
Completo	91,5	94,4	87,2	93,6	89,0	88,0	92,6	92,7	84,4	90,4
Média	86,9			90,6			91,9			-

Uma primeira constatação que se faz é que com o modelo TinyLlama, o ajuste fino completo não apresentou vantagem em relação as outras abordagens, embora seja válido questionar pelo tamanho maior do modelo se seria necessário mais épocas para este atingir o seu melhor resultado possível.

A acurácia média cresce à medida que o modelo de linguagem vai se tornando mais robusto. Entretanto, não se observa um ganho significativo entre o uso do RoBERTa e o TinyLlama em vários casos, considerando que o primeiro modelo é quase dez vezes menor, a diferença da acurácia obtida, para os testes aqui realizados, é pequena, sugerindo que um modelo de linguagem menor deve ser considerado como uma opção para problemas de classificação de texto. Por outro lado fazendo um comparativo entre DistilBERT e RoBERTa, nota-se claramente uma grande vantagem para o RoBERTa na maior parte dos testes.

Observou-se que a técnica do LoRA obteve o melhor resultado entre as técnicas de ajuste fino eficientes, em praticamente todos os testes realizados, e em conjunto com o

TinyLlama apresentou a melhor combinação para obtenção de acurácia. Era esperado que o AdaLoRA tivesse uma performance pelo menos levemente superior ao LoRA, visto que o objetivo da técnica é ser um avanço do LoRA ao tentar focar nas matrizes mais importantes do modelo, mas ocorreu exatamente o contrário tendo o AdaLoRA uma acurácia média de 0.5% pior que o LoRA diferente dos resultados obtidos pelo trabalhando de Zhang e outros [66], onde o AdaLoRA foi melhor.

O ajuste fino completo teve acurácia média geral igual ao do AdaLoRA, e pouco pior que o LoRA. No DistilBERT o ajuste fino completo teve resultados consideravelmente melhores que técnicas de ajuste fino eficientes. Já no RoBERTa e TinyLlama o ajuste fino completo teve acurácia próxima ou pior (maior parte dos casos) do que o LoRA e AdaLoRA. Reforçando a tendencia de que quanto maior o modelo mais eficaz é o uso de técnicas de ajuste fino eficientes.

O IA3 teve claramente uma performance abaixo das outras técnicas na maior parte dos testes.

Os testes aqui realizados sugere que os modelos de linguagem com menor quantidade de parâmetros, a exemplo do RoBERTa, podem ser preferencialmente utilizados em cenários que possuem alguma restrição de recursos, visto que as diferenças de acurácias não são tão significativas. O custo do incremento de processamento pode não compensar o ganho de acurácia para diversas aplicações, particularmente para as testadas neste trabalho.

4.4 Resultados e Análises do processo de aprendizado ocorrido nos experimentos

Esta seção analisa o comportamento da evolução do processo de treinamento. Os subsídios para essa análise estão apresentados nas Tabelas 4.4, 4.5 e 4.6. Elas apresentam o valor da acurácia das massas de treino e de teste, para cada uma das três épocas utilizadas no treinamento. As tabelas apresentam os resultados para os modelos DistilBERT, RoBERTa e TinyLlama respectivamente.

Tabela 4.4: Resultados das acurácias obtidas nos dados de treino e teste por épocas no modelo DistilBERT.

FN	IMDb - Acc. Train/Test (%)			AG News - Acc. Train/Test (%)			SNLI - Acc. Train/Test (%)		
	Epoch 1	Epoch 2	Epoch 3	Epoch 1	Epoch 2	Epoch 3	Epoch 1	Epoch 2	Epoch 3
LoRA	87.5/87.4	88.6/88.7	88.8/88.8	91.1/90.7	92.0/91.6	92.4/91.8	78.0/79.2	80.0/80.7	80.6/81.3
AdaLoRA	85.1/85.1	86.8/86.8	87.1/87.2	90.4/90.1	91.0/90.7	91.1/90.8	77.6/79.1	79.8/81.0	80.5/81.4
IA3	83.7/83.4	84.8/84.6	85.2/85.0	89.6/89.4	90.0/89.8	90.1/89.8	68.7/70.4	72.0/73.3	73.0/74.1
Completo	94.7/90.5	98.4/91.4	99.4/91.5	94.9/93.0	97.1/93.8	98.0/94.4	85.1/84.4	88.7/86.0	91.4/87.2

Observando-se a Tabela 4.4, pode-se notar que para o modelo de linguagem DistilBERT, as técnicas de ajuste fino eficientes apresentam um processo de aprendizado e de generalização bem sincronizado, indicando que o treinamento está sendo feito com qualidade, sem nenhum traço de ocorrência de *overfitting*. Pode-se concluir que tais técnicas, devido ao baixo número de parâmetros modificados, realmente contribuem para a redução da possibilidade de *overfitting* ou *catastrophic forgetting*. Considerando que tais técnicas não alteram os parâmetros pré-treinados do modelo original, pode-se considerar que abordagens como essas tendem a serem cada vez mais utilizadas, devido a essas vantagens já conhecidas e apresentadas nos testes aqui realizados.

Tabela 4.5: Resultados das acurácias obtidas nos dados de treino e teste por épocas no modelo RoBERTa.

Ajuste fino	IMDb - Acc. Train/Test (%)			AG News - Acc. Train/Test (%)			SNLI - Acc. Train/Test (%)		
	Epoch 1	Epoch 2	Epoch 3	Epoch 1	Epoch 2	Epoch 3	Epoch 1	Epoch 2	Epoch 3
LoRA	93,3/93,6	93,6/94,0	93,9/94,3	92,5/92,2	93,4/93,1	93,6/93,2	85,7/87,2	86,5/88,0	86,7/88,1
AdaLoRA	92,5/92,8	93,1/93,3	93,3/93,6	92,1/92,0	92,8/92,5	93,1/92,7	85,3/87,2	86,2/87,8	86,4/87,9
IA3	89,1/89,3	90,2/90,6	90,7/90,9	91,1/90,9	91,5/91,3	91,6/91,4	79,3/80,7	81,6/83,2	82,1/83,9
Completo	91,6/90,3	95,3/92,7	97,1/93,6	87,5/87,4	88,2/88,1	89,3/89,0	84,8/87,1	86,0/87,1	87,2/88,0

O mesmo comportamento observado com o modelo DistilBERT é também observado com o RoBERTa, valendo assim todas as análises realizadas para o primeiro modelo apresentado. Até o comportamento de distanciamento entre a acurácia de massa treino com a de teste para o ajuste fino completo é observada.

A similaridade do uso dos dois modelos citados anteriormente, sugerem uma certa equivalência dos modelos de linguagem, naturalmente, para os problemas tratados neste trabalho.

Tabela 4.6: Resultados das acurácias obtidas nos dados de treino e teste por épocas no modelo TinyLlama.

FN	IMDb - Acc. Train/Test (%)			AG News - Acc. Train/Test (%)			SNLI - Acc. Train/Test (%)		
	Epoch 1	Epoch 2	Epoch 3	Epoch 1	Epoch 2	Epoch 3	Epoch 1	Epoch 2	Epoch 3
LoRA	95,7/95,7	96,2/95,9	96,4/96,1	94,0/93,4	94,8/93,9	95,1/94,0	88,8/90,1	89,2/90,5	89,9/90,9
AdaLoRA	94,7/95,0	95,3/95,6	95,4/95,7	92,7/92,1	93,6/93,2	93,8/93,2	88,1/89,2	89,2/90,1	90,0/91,1
IA3	89,6/89,4	92,1/92,1	92,6/92,6	91,8/91,4	92,5/92,1	92,8/92,3	81,2/82,3	83,3/83,9	87,2/87,6
Completo	92,4/89,9	97,1/92,1	99,6/92,6	92,4/90,5	95,4/92,3	96,8/92,7	76,3/75,0	79,1/79,3	83,5/84,4

O mesmo comportamento observado com o modelo DistilBERT e RoBERTa mais uma vez se repete com o TinyLlama, mantendo uma validade das análises anteriores para este último modelo de linguagem testado.

4.5 Resultados e Análises do quantitativo de parâmetros treináveis

A Tabela 4.7 mostra o total de parâmetros ajustáveis em cada uma das técnicas de ajuste fino. A partir dessa análise é possível perceber a robustez do processo, bem como a estabilidade do conhecimento inserido na fase de pré-treinamento.

Tabela 4.7: Quantidade de parâmetros passíveis de ajuste em cada técnica de ajuste fino

Técnica	DistilBERT		RoBERTa		TinyLlama	
	Parâmetros	% do total	Parâmetros	% do total	Parâmetros	% do total
LoRA	665,858	1.0%	739,586	0.59%	567,472	0.0548%
AdaLoRA	665,906	1.0%	739,682	0.59%	567,296	0.0548%
IA3	601,346	0.9%	610,562	0.49%	504,632	0.0487%
Completo	66,448,900	100.0%	125,386,756	100.0%	1,035,083,776	100.0%

Observa-se na tabela que todas as técnicas de ajuste fino eficiente alteram um percentual muito pequeno dos parâmetros que irão compor o modelo final. A robustez da técnica quanto a manter o conhecimento do modelo pré-treinado pode ser perfeitamente notado na tabela, onde essas técnicas alteram um pequeno percentual dos parâmetros do modelo. Observa-se que quão maior o tamanho do modelo de linguagem, menor é o percentual dos parâmetros ajustáveis. O menor percentual de parâmetros modificados ocorreu na técnica IA3 no modelo TinyLlama, sendo ele de apenas 0,049%. Isso corrobora com conceitos já discutidos nesse trabalho, como uma menor capacidade de gerar *overfitting*, causar *catastrophic forgetting* e economia de memória.

4.6 Contribuições do trabalho

O trabalho realizado traz contribuições, no contexto da eficiência de modelos de linguagem natural. A pesquisa investigou o desempenho de três modelos distintos – DistilBERT, RoBERTa e TinyLLaMA – aplicando quatro técnicas diferentes de ajuste fino: LoRA, AdaLoRA, IA3 e ajuste fino completo. Os modelos foram treinados e avaliados utilizando três conjuntos de dados amplamente reconhecidos: IMDb, AG News e SNLI. A análise focou em três métricas principais: consumo de memória, tempo de treinamento e acurácia.

4.6.1 Principais Contribuições

Avaliação de Modelos com Diferentes Capacidades

A comparação entre DistilBERT, RoBERTa e TinyLLaMA permitiu identificar como modelos de diferentes tamanhos e arquiteturas se comportam diante das mesmas tarefas. Esta análise é crucial para a escolha do modelo mais adequado, considerando os recursos disponíveis e os requisitos de desempenho.

Eficácia das Técnicas de Ajuste Fino

Ao aplicar LoRA, AdaLoRA, IA3 e o ajuste fino completo, a pesquisa fornece uma visão clara sobre a eficiência de cada técnica. Isso é especialmente útil para a comunidade científica e para profissionais que buscam otimizar modelos sem comprometer a qualidade, economizando tempo e recursos computacionais.

Comparação de Desempenho em Diferentes Conjuntos de Dados

A utilização dos conjuntos de dados IMDb, AG News e SNLI permitiu testar os modelos em diferentes tipos de tarefas de linguagem natural: classificação de sentimentos, classificação de notícias e inferência lógica. Isso demonstra a versatilidade e os limites de cada técnica e modelo em cenários variados.

Análise de Consumo de Memória e Tempo de Treinamento

Um dos aspectos críticos abordados foi o consumo de memória e o tempo de treinamento, métricas fundamentais para a aplicação prática de modelos de IA. A pesquisa identifica quais combinações de modelo e técnica são mais eficientes, fornecendo diretrizes claras para a implementação em ambientes com restrições de recursos.

Impacto na Acurácia

A acurácia é uma métrica essencial para avaliar a eficácia dos modelos. A comparação detalhada entre as diferentes abordagens permitiu identificar quais técnicas de ajuste fino mantêm ou melhoram a precisão, oferecendo *insights* sobre a viabilidade de reduzir o tamanho do modelo sem sacrificar a performance.

Considerações Finais

O trabalho contribui de forma abrangente para o entendimento de como diferentes modelos de linguagem natural e técnicas de ajuste fino podem ser otimizados para atender a diversas necessidades práticas. A pesquisa destaca que não há uma solução única

que seja superior em todos os aspectos, mas sim que a escolha do modelo e da técnica de ajuste deve ser orientada pelos requisitos específicos de cada aplicação, considerando o equilíbrio entre consumo de memória, tempo de treinamento e acurácia.

CAPÍTULO 5

Conclusões e Trabalhos Futuros

5.1 Conclusões

Com relação aos objetivos do projeto, todos estudos necessários para a realização do trabalho estão resumidamente apresentados no capítulo 2. Os demais objetivos específicos foram atingidos pela materialização deste documento, particularmente os Capítulos 3 e 4 que reportam o planejamento, a implementação e os resultados dos experimentos. Assim, por atingir com êxito os objetivos específicos, conclui-se que o objetivo geral do projeto foi atingido. Com relação às análises foco deste trabalho, que são tempo de processamento, consumo de memória e acurácia obtida, conclui-se que:

- Quanto ao consumo de memória:
  1. As técnicas de ajuste fino eficientes realmente utilizam um montante menor de memória que o ajuste fino completo
  2. As técnicas de ajuste fino eficientes mantiveram uma utilização de memória praticamente iguais, sendo o LoRA ligeiramente maior, cerca de 10% em alguns casos, que as outras.
  3. O tamanho da base de dados não influiu no uso de memória, mas o modelo de linguagem teve muita influência no uso da memória, exceto na combinação TinyLlama e SNLI.
  4. O uso de memória, para as técnicas eficientes testadas, não é um critério decisivo para a escolha de uma técnica avançada de ajuste fino.
- Quanto ao tempo de processamento:
  1. O tempo de processamento do experimento está ligado principalmente ao modelo de linguagem e ao tamanho do banco de dados de teste.
  2. As técnicas apresentadas mostram pequenas variações de tempo, sendo que o ajuste fino completo foi quase sempre o modelo mais demorado.
  3. Observou-se que o tempo de processamento cresce de forma não linear quando se aumenta o número de parâmetros do modelo de linguagem.

4. O tempo de processamento, para as técnicas testadas, também não é um critério decisivo para a escolha de uma técnica de ajuste fino.

- Quanto à acurácia:
  1. Os resultados da acurácia apresentaram diferenças importantes entre as técnicas, visto que um ponto percentual é importante, dependendo da aplicação.
  2. As técnicas LoRA e AdaLoRA Apresentaram resultados semelhantes, sendo que na média o LoRA superou o AdaLoRA. O IA3 não superou as outras técnicas avançadas em nenhum dos testes.
  3. O LoRA superou o ajuste fino completo nos modelos RoBERTa e TinyLlama, se mostrando como a melhor técnica analisada.
  4. Observou-se uma tendência de que quanto maior o modelo, mais eficaz é o uso de técnicas de ajuste fino eficientes.

Finalmente, o trabalho mostrou que alguns resultados que destoaram do esperado, tais como:

1. O AdaLoRA que é uma melhoria do LoRA, em geral teve resultados piores que o LoRA e teve na maior parte dos testes um tempo de treinamento superior do ajuste fino completo.
2. Não se observou nos experimentos todas as vantagens descritas para as técnicas nos trabalhos que a propuseram. Mais testes precisam ser feitos para avaliar as razões dessas discrepâncias.

5.2 Trabalhos Futuros

As técnicas de ajuste fino com eficiência de parâmetros testadas tendem a ter melhores resultados em modelos maiores, entretanto por limitações de tempo e hardware o maior modelo testado foi o TinyLlama com aproximadamente 1.1 bilhões de parâmetros, e o próprio modelo llama possui versões com 7, 13 e até 70 bilhões de parâmetros. Então seria produtivo fazer testes em modelos de escala maiores também tendo em vista que quanto maior o modelo maior é o problema do consumo de memória e tempo de treinamento.

Todos os ajustes finos e datasets testados neste trabalho são relacionados a classificação de texto, e para um melhor comparativo é importante realizar testes com outros tipos de tarefas específicas em PLN, como por exemplo tradução automática e reconhecimento de entidades nomeadas.

Para trabalhos futuros proponho uma divisão em duas etapas. A primeira é utilizar os mesmos modelos utilizados neste trabalhos mas executando ajustes finos com

um conjunto de dados relacionado a tradução automática de textos e outro relacionado a reconhecimento de entidades nomeadas, com isso seria possível tirar conclusões mais precisas já que o total de tarefas específicas diferentes seria três em vez de uma.

A segunda etapa seria utilizar todos os conjunto de dados utilizados neste trabalho e também os da primeira etapa, totalizando sete conjuntos de dados diferentes divididos em três tarefas específicas, mas dessa vez utilizando o llama de 7 e 13 bilhões de parâmetros. Neste caso teremos testes utilizando modelos relativamente grandes para o cenário atual e que ainda são viáveis de serem executados em uma GPU A100.

Com todos esses trabalhos finalizados, teremos um comparativo com modelos de tamanhos muito diferentes, sendo eles: DistilBERT (66M), RoBERTa (125M), TinyLlama (1B), Llama-7B (7B), Llama-13B (13B). E como dito anteriormente serão sete conjuntos de dados diferentes divididos em três tarefas específicas. Todas essas combinações teriam o ajuste fino feito com o LoRA, AdaLoRA, IA3 e ajuste fino completo. Com isso sendo possível analisar vantagens e desvantagens de cada uma das combinações para diferentes cenários de restrição de memória, tempo de processamento e necessidade de acurácia.

Referências Bibliográficas

[1] AGHAJANYAN, A.; ZETTLEMOYER, L.; GUPTA, S. **Intrinsic dimensionality explains the effectiveness of language model fine-tuning**. *arXiv preprint arXiv:2012.13255*, 2020.

[2] AI, G. **Palm 2**.

[3] AI, M. **Discover more possibilities, learn and create with your own intelligent assistant**.

[4] BENTON, G.; MADDOX, W.; WILSON, A. G. **Rethinking parameter counting: Effective dimensionality revisited**. *arXiv*, 2020.

[5] BOWMAN, S. R.; ANGELI, G.; POTTS, C.; MANNING, C. D. **A large annotated corpus for learning natural language inference**. *arXiv preprint arXiv:1508.05326*, 2015.

[6] BROWN, T.; MANN, B.; RYDER, N.; SUBBIAH, M.; KAPLAN, J. D.; DHARIWAL, P.; NEELAKANTAN, A.; SHYAM, P.; SASTRY, G.; ASKELL, A.; OTHERS. **Language models are few-shot learners**. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[7] CHURCH, K. W. **Word2vec**. *Natural Language Engineering*, 23(1):155–162, 2017.

[8] CORPORATION, N. **Python bindings for the nvidia management library**.

[9] DAI, Z.; YANG, Z.; YANG, Y.; CARBONELL, J.; LE, Q. V.; SALAKHUTDINOV, R. **Transformer-xl: Attentive language models beyond a fixed-length context**, 2019.

[10] DAO, T.; FU, D.; ERMON, S.; RUDRA, A.; RÉ, C. **Flashattention: Fast and memory-efficient exact attention with io-awareness**. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

[11] DATABASE, T. I. M. **Imdb dataset**.

[12] DEVLIN, J.; CHANG, M.-W.; LEE, K.; TOUTANOVA, K. **Bert: Pre-training of deep bidirectional transformers for language understanding**. *arXiv preprint arXiv:1810.04805*, 2018.

[13] DING, N.; QIN, Y.; YANG, G.; WEI, F.; YANG, Z.; SU, Y.; HU, S.; CHEN, Y.; CHAN, C.-M.; CHEN, W.; OTHERS. **Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models**. *arXiv preprint arXiv:2203.06904*, 2022.

[14] FU, Z.; YANG, H.; SO, A. M.-C.; LAM, W.; BING, L.; COLLIER, N. **On the effectiveness of parameter-efficient fine-tuning**. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, p. 12799–12807, 2023.

[15] HINTON, G.; VINYALS, O.; DEAN, J. **Distilling the knowledge in a neural network**. *arXiv preprint arXiv:1503.02531*, 2015.

[16] HOFFMANN, J.; NAVARRO, O.; KASTNER, F.; JANSSEN, B.; HUBNER, M. **A survey on cnn and rnn implementations**. In: *PESARO 2017: The Seventh International Conference on Performance, Safety and Robustness in Complex Systems and Applications*, número 3, 2017.

[17] HOULSBY, N.; GIURGIU, A.; JASTRZEBSKI, S.; MORRONE, B.; DE LAROUSSILHE, Q.; GESMUNDO, A.; ATTARIYAN, M.; GELLY, S. **Parameter-efficient transfer learning for nlp**. In: *International conference on machine learning*, p. 2790–2799. PMLR, 2019.

[18] HOWARD, J.; RUDER, S. **Universal language model fine-tuning for text classification**. *arXiv preprint arXiv:1801.06146*, 2018.

[19] HU, E. J.; SHEN, Y.; WALLIS, P.; ALLEN-ZHU, Z.; LI, Y.; WANG, S.; WANG, L.; CHEN, W. **Lora: Low-rank adaptation of large language models**. *arXiv preprint arXiv:2106.09685*, 2021.

[20] HUGGINGFACE COMMUNITY. **The ai community building the future**.

[21] IMAN, M.; ARABNIA, H. R.; RASHEED, K. **A review of deep transfer learning and recent advancements**. *Technologies*, 11(2):40, 2023.

[22] JOHRI, P.; KHATRI, S. K.; AL-TAANI, A. T.; SABHARWAL, M.; SUVANOV, S.; KUMAR, A. **Natural language processing: History, evolution, application, and future work**. In: *Proceedings of 3rd International Conference on Computing Informatics and Networks: ICCIN 2020*, p. 365–375. Springer, 2021.

[23] JUSOH, S. **A study on nlp applications and ambiguity problems**. *Journal of Theoretical and Applied Information Technology*, 96:1486–1499, 03 2018.

[24] KATARIA, B.; JETHVA, H. B. **Cnn-bidirectional lstm based optical character recognition of sanskrit manuscripts: A comprehensive systematic literature review**. *Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol.(IJSRCSEIT)*, 5(2):2456–3307, 2019.

[25] KHURANA, D.; KOLI, A.; KHATTER, K.; SINGH, S. **Natural language processing: State of the art, current trends and challenges**. *Multimedia tools and applications*, 82(3):3713–3744, 2023.

[26] KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. **Imagenet classification with deep convolutional neural networks**. *Advances in neural information processing systems*, 25, 2012.

[27] LE, Q.; MIKOLOV, T. **Distributed representations of sentences and documents**. *31st International Conference on Machine Learning, ICML 2014*, 4, 05 2014.

[28] LI, C.; FARKHOOR, H.; LIU, R.; YOSINSKI, J. **Measuring the intrinsic dimension of objective landscapes**. *arXiv preprint arXiv:1804.08838*, 2018.

[29] LIALIN, V.; DESHPANDE, V.; RUMSHISKY, A. **Scaling down to scale up: A guide to parameter-efficient fine-tuning**. *arXiv preprint arXiv:2303.15647*, 2023.

[30] LIU, H.; TAM, D.; MUQEETH, M.; MOHTA, J.; HUANG, T.; BANSAL, M.; RAFFEL, C. A. **Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning**. *Advances in Neural Information Processing Systems*, 35:1950–1965, 2022.

[31] LIU, Y.; OTT, M.; GOYAL, N.; DU, J.; JOSHI, M.; CHEN, D.; LEVY, O.; LEWIS, M.; ZETTLEMOYER, L.; STOYANOV, V. **Roberta: A robustly optimized bert pretraining approach**. *arXiv preprint arXiv:1907.11692*, 2019.

[32] LUCCIONI, A. S.; HERNANDEZ-GARCIA, A. **Counting carbon: A survey of factors influencing the emissions of machine learning**. *arXiv preprint arXiv:2302.08476*, 2023.

[33] MALTE, A.; RATADIYA, P. **Evolution of transfer learning in natural language processing**, 2019.

[34] MANN, B.; RYDER, N.; SUBBIAH, M.; KAPLAN, J.; DHARIWAL, P.; NEELAKANTAN, A.; SHYAM, P.; SASTRY, G.; ASKELL, A.; AGARWAL, S.; OTHERS. **Language models are few-shot learners**. *arXiv preprint arXiv:2005.14165*, 2020.

[35] MIKOLOV, T.; CHEN, K.; CORRADO, G.; DEAN, J. **Efficient estimation of word representations in vector space**. *Proceedings of Workshop at ICLR*, 2013, 01 2013.

[36] MIKOLOV, T.; SUTSKEVER, I.; CHEN, K.; CORRADO, G.; DEAN, J. **Distributed representations of words and phrases and their compositionality**, 2013.

[37] MIKOLOV, T.; ZWEIG, G. **Context dependent recurrent neural network language model**. In: *2012 IEEE Spoken Language Technology Workshop (SLT)*, p. 234–239, 2012.

[38] O'SHEA, K.; NASH, R. **An introduction to convolutional neural networks**. *CoRR*, abs/1511.08458, 2015.

[39] PATTERSON, D.; GONZALEZ, J.; HÖLZLE, U.; LE, Q.; LIANG, C.; MUNGUIA, L.-M.; ROTHCHILD, D.; SO, D.; TEXIER, M.; DEAN, J. **The carbon footprint of machine learning training will plateau, then shrink**, 2022.

[40] PEREZ, E.; KIELA, D.; CHO, K. **True few-shot learning with language models**. *Advances in neural information processing systems*, 34:11054–11070, 2021.

[41] PETERS, M. E.; NEUMANN, M.; IYYER, M.; GARDNER, M.; CLARK, C.; LEE, K.; ZETTLEMOYER, L. **Deep contextualized word representations**, 2018.

[42] RADFORD, A.; NARASIMHAN, K.; SALIMANS, T.; SUTSKEVER, I. **Improving language understanding by generative pre-training**, 2018.

[43] RADFORD, A.; NARASIMHAN, K. **Improving language understanding by generative pre-training**, 2018.

[44] RADFORD, A.; WU, J.; CHILD, R.; LUAN, D.; AMODEI, D.; SUTSKEVER, I.; OTHERS. **Language models are unsupervised multitask learners**. *OpenAI blog*, 1(8):9, 2019.

[45] RAFFEL, C.; SHAZEER, N.; ROBERTS, A.; LEE, K.; NARANG, S.; MATENA, M.; ZHOU, Y.; LI, W.; LIU, P. J. **Exploring the limits of transfer learning with a unified text-to-text transformer**. *Journal of machine learning research*, 21(140):1–67, 2020.

[46] ROKH, B.; AZARPEYVAND, A.; KHANTEYMOORI, A. **A comprehensive survey on model quantization for deep neural networks**. *arXiv preprint arXiv:2205.07877*, 2022.

[47] SAHOO, P.; SINGH, A. K.; SAHA, S.; JAIN, V.; MONDAL, S.; CHADHA, A. **A systematic survey of prompt engineering in large language models: Techniques and applications**, 2024.

[48] SANH, V.; DEBUT, L.; CHAUMOND, J.; WOLF, T. **Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter**. *arXiv preprint arXiv:1910.01108*, 2019.

[49] SANH, V.; WEBSON, A.; RAFFEL, C.; BACH, S. H.; SUTAWIKA, L.; ALYAFEAI, Z.; CHAFFIN, A.; STIEGLER, A.; SCAO, T. L.; RAJA, A.; OTHERS. **Multitask prompted training enables zero-shot task generalization**. *arXiv preprint arXiv:2110.08207*, 2021.

[50] SENNRICH, R.; ZHANG, B. **Revisiting low-resource neural machine translation: A case study**, 2019.

[51] SHERSTINSKY, A. **Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network**. *CoRR*, abs/1808.03314, 2018.

[52] SHIN, J.; TANG, C.; MOHATI, T.; NAYEBI, M.; WANG, S.; HEMMATI, H. **Prompt engineering or fine tuning: An empirical assessment of large language models in automated software engineering tasks**. *arXiv preprint arXiv:2310.10508*, 2023.

[53] SU, J.; LU, Y.; PAN, S.; MURTADHA, A.; WEN, B.; LIU, Y. **Roformer: Enhanced transformer with rotary position embedding**, 2023.

[54] SUNDERMEYER, M.; SCHLUTER, R.; NEY, H. **Lstm neural networks for language modeling**. In: *Proceedings of the 13th Annual Conference of the International Speech Communication Association*, p. 194–201, 2012.

[55] TOUVRON, H.; LAVRIL, T.; IZACARD, G.; MARTINET, X.; LACHAUX, M.-A.; LACROIX, T.; ROZIÈRE, B.; GOYAL, N.; HAMBRO, E.; AZHAR, F.; OTHERS. **Llama: Open and efficient foundation language models**. *arXiv preprint arXiv:2302.13971*, 2023.

[56] TREVISO, M.; LEE, J.-U.; JI, T.; AKEN, B. V.; CAO, Q.; CIOSICI, M. R.; HASSID, M.; HEAFIELD, K.; HOOKER, S.; RAFFEL, C.; OTHERS. **Efficient methods for natural language processing: A survey**. *Transactions of the Association for Computational Linguistics*, 11:826–860, 2023.

[57] VADERA, S.; AMEEN, S. **Methods for pruning deep neural networks**. *IEEE Access*, 10:63280–63300, 2022.

[58] VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, Ł.; POLOSUKHIN, I. **Attention is all you need**. *Advances in neural information processing systems*, 30, 2017.

[59] VU, T.; LESTER, B.; CONSTANT, N.; AL-RFOU, R.; CER, D. **Spot: Better frozen model adaptation through soft prompt transfer**. *arXiv preprint arXiv:2110.07904*, 2021.

[60] WANG, H.; LI, J.; WU, H.; HOVY, E.; SUN, Y. **Pre-trained language models and their applications**. *Engineering*, 2022.

[61] WELLECK, S.; KULIKOV, I.; ROLLER, S.; DINAN, E.; CHO, K.; WESTON, J. **Neural text generation with unlikelihood training**. *arXiv preprint arXiv:1908.04319*, 2019.

[62] WU, Z.; ARORA, A.; WANG, Z.; GEIGER, A.; JURAFSKY, D.; MANNING, C. D.; POTTS, C. **Reft: Representation finetuning for language models**. *arXiv preprint arXiv:2404.03592*, 2024.

[63] XU, L.; XIE, H.; QIN, S.-Z. J.; TAO, X.; WANG, F. L. **Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment**. *arXiv preprint arXiv:2312.12148*, 2023.

[64] YANG, Z.; DAI, Z.; YANG, Y.; CARBONELL, J.; SALAKHUTDINOV, R.; LE, Q. V. **Xlnet: Generalized autoregressive pretraining for language understanding**, 2020.

[65] ZHANG, P.; ZENG, G.; WANG, T.; LU, W. **Tinyllama: An open-source small language model**. *arXiv preprint arXiv:2401.02385*, 2024.

[66] ZHANG, Q.; CHEN, M.; BUKHARIN, A.; HE, P.; CHENG, Y.; CHEN, W.; ZHAO, T. **Adaptive budget allocation for parameter-efficient fine-tuning**. In: *The Eleventh International Conference on Learning Representations*, 2022.

[67] ZHANG, X.; ZHAO, J.; LECUN, Y. **Character-level convolutional networks for text classification**. *Advances in neural information processing systems*, 28, 2015.

[68] ZHAO, H.; CHEN, H.; YANG, F.; LIU, N.; DENG, H.; CAI, H.; WANG, S.; YIN, D.; DU, M. **Explainability for large language models: A survey**. *ACM Transactions on Intelligent Systems and Technology*, 15(2):1–38, 2024.

[69] ZHAO, W. X.; ZHOU, K.; LI, J.; TANG, T.; WANG, X.; HOU, Y.; MIN, Y.; ZHANG, B.; ZHANG, J.; DONG, Z.; OTHERS. **A survey of large language models**. *arXiv preprint arXiv:2303.18223*, 2023.

[70] ZHUANG, F.; QI, Z.; DUAN, K.; XI, D.; ZHU, Y.; ZHU, H.; XIONG, H.; HE, Q. **A comprehensive survey on transfer learning**. *Proceedings of the IEEE*, 109(1):43–76, 2020.