TPF-03 Redes Neurais

Juan Avelar - 4229
Ciência da Computação
Universidade Federal de Viçosa
Campus Florestal
juan.avelar@ufv.br

1 Introdução

O presente trabalho dá continuidade à reprodução experimental do artigo *"DistilBERT, a distilled version of BERT"* (Sanh et al., 2019). Na etapa anterior (TPF-02), reproduziu-se o *fine-tuning* padrão do modelo no dataset SST-2 (*Stanford Sentiment Treebank*), estabelecendo uma linha de base robusta de desempenho.

O objetivo desta fase (TPF-03) é investigar estratégias que possam melhorar o desempenho do modelo ou sua eficiência de treinamento. Modelos baseados em Transformers, embora poderosos, demandam alto custo computacional. Portanto, propõem-se duas intervenções distintas: uma alteração arquitetural via congelamento de pesos (*Layer Freezing*) visando eficiência, e um refinamento do procedimento de otimização através de novos hiperparâmetros visando aprimorar a convergência.

2 Modificações e Justificativas

Para superar a reprodução base, foram selecionadas duas abordagens tecnicamente distintas descritas abaixo.

2.1 Modificação 1: Congelamento Parcial de Camadas (Layer Freezing)

Diferente da implementação original onde todos os parâmetros (cerca de 66 milhões) são atualizados durante o *fine-tuning*, nesta abordagem congelamos os pesos da camada de *embeddings* e das três primeiras camadas do codificador Transformer (camadas 0, 1 e 2). Apenas as três últimas camadas (3, 4, 5) e o cabeçote de classificação (*classifier head*) permaneceram treináveis.

Implementação: O código abaixo ilustra como os gradientes foram desativados para as camadas selecionadas utilizando a biblioteca PyTorch/Transformers.

Listing 1: Código para Congelamento de Camadas

```
def model_init_frozen():
    model = AutoModelForSequenceClassification.from_pretrained(
        MODEL_CHECKPOINT, num_labels=2
    )

    # Congelar Embeddings
    for param in model.distilbert.embeddings.parameters():
        param.requires_grad = False

    # Congelar as 3 primeiras camadas (0, 1, 2) do encoder
    for i in range(3):
        for param in model.distilbert.transformer.layer[i].parameters():
            param.requires_grad = False
```

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

```
    return model
```

Justificativa: Modelos pré-treinados como o DistilBERT capturam características linguísticas fundamentais (sintaxe, morfologia) nas camadas iniciais, que são altamente transferíveis entre tarefas. Ao congelá-las, reduzimos drasticamente o número de gradientes a serem calculados, diminuindo o uso de memória da GPU e o tempo de treinamento, além de evitar o *esquecimento catastrófico* de *features* de baixo nível.

2.2 Modificação 2: Otimização de Hiperparâmetros

A segunda modificação mantém a arquitetura original totalmente descongelada, mas altera a dinâmica de otimização. Substituímos o agendador de taxa de aprendizado linear (padrão do Hugging Face) por um agendador *Cosine with Warmup* e aumentamos a regularização *Weight Decay* de 0.01 para 0.05. A taxa de aprendizado inicial também foi levemente ajustada para $3e^{-5}$.

Implementação: Os novos hiperparâmetros foram definidos no objeto `TrainingArguments` conforme abaixo:

Listing 2: Novos Hiperparâmetros

```
training_args_mod2 = {
    "learning_rate": 3e-5,          # Aumento leve na LR
    "weight_decay": 0.05,           # Maior regularizacao
    "lr_scheduler_type": "cosine", # Scheduler cosseno
    "warmup_ratio": 0.1             # 10% de warmup
}
```

Justificativa: O agendador linear tende a reduzir a taxa de aprendizado de forma monótona e por vezes abrupta. O decaimento cosseno oferece uma redução mais suave, permitindo que o modelo explore melhor o espaço de perda nas etapas finais. O aumento do *weight decay* impõe uma penalidade maior à complexidade dos pesos, buscando melhorar a generalização no conjunto de teste/validação.

3 Configuração Experimental e Resultados

3.1 Dataset e Configuração

O dataset utilizado foi o SST-2 (*Stanford Sentiment Treebank*), parte do benchmark GLUE. O conjunto de dados é composto por **67.349 sentenças de treinamento**, **872 sentenças de validação** (dev set) e **1.821 sentenças de teste**. A métrica de avaliação principal é a acurácia.

Os experimentos foram realizados utilizando a biblioteca `transformers` da Hugging Face em ambiente acelerado por GPU (Google Colab). O dataset utilizado foi o SST-2 (benchmark GLUE). Todos os modelos foram treinados por 3 épocas.

3.2 Resultados Obtidos

A Tabela 1 resume a comparação entre a reprodução original (Baseline) e as duas modificações propostas.

Table 1: Comparação de Desempenho e Eficiência no Dataset SST-2 (Dev Set)

Experimento	Acurácia	Loss	Tempo de Treino	Ganho de Tempo
Reprodução Original	**90.71%**	0.3255	$\approx$ 22 min 51s	-
Mod 1: Layer Freezing	90.02%	0.3330	$\approx$ **14 min 48s**	**+35%**
Mod 2: Hyperparam Opt	90.60%	0.3804	$\approx$ 22 min 46s	0%

3.3 Análise

O modelo original atingiu a maior acurácia (90.71%), servindo como uma baseline extremamente competitiva. A Modificação 1 (*Layer Freezing*) resultou em uma redução significativa do tempo de treinamento (de 23min para 15min), com uma queda marginal de desempenho de apenas 0.69 pontos percentuais. A Modificação 2 obteve um resultado estatisticamente similar à baseline (90.60%), indicando que o agendador cosseno manteve a estabilidade, mas não foi suficiente para superar os parâmetros originais já otimizados pelos autores do DistilBERT.

4 Discussão e Conclusões

Os experimentos conduzidos neste trabalho demonstraram que a eficiência computacional pode ser significativamente aprimorada com impacto mínimo na qualidade preditiva.

A principal contribuição deste estudo foi a validação da **Modificação 1 (Congelamento de Camadas)**. Ao congelar a metade inferior do modelo, conseguimos uma aceleração de 35% no processo de *fine-tuning*. No contexto atual de LLMs, onde o custo de GPU é o principal gargalo, sacrificar menos de 1% de acurácia por um ganho de velocidade dessa magnitude representa um *trade-off* altamente vantajoso, especialmente para prototipagem rápida ou implantação em dispositivos de borda (*edge devices*).

Em relação à **Modificação 2**, observou-se que os hiperparâmetros originais propostos por Sanh et al. (2019) são robustos para tarefas do GLUE. Embora o agendador cosseno seja teoricamente superior em muitos cenários, para o SST-2 (que converge rapidamente em 3 épocas), a diferença prática foi negligenciável. Isso sugere que a arquitetura do DistilBERT é resiliente a variações moderadas de hiperparâmetros.

Conclui-se que o projeto TPF-03 cumpriu seus objetivos de reprodução e investigação, destacando que técnicas de eficiência como o *freezing* são ferramentas vitais para democratizar o acesso ao treinamento de Transformers.

5 Link do Repositório

`https://github.com/juan-avelar/RedesNeurais`

Referências Bibliográficas

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*. arXiv preprint arXiv:1910.01108.