# **Análise Comparativa de Modelos de Linguagem Baseados em Transformers: Metodologias e Parâmetros de Treinamento**

**1. Introdução**

A introdução da arquitetura Transformer 1 representou um ponto de inflexão no campo do Processamento de Linguagem Natural (PLN). Subsequentemente, o modelo BERT (Bidirectional Encoder Representations from Transformers) 4 consolidou essa revolução ao introduzir o pré-treinamento bidirecional profundo. Essa abordagem permitiu que modelos aprendessem representações contextuais ricas a partir de grandes volumes de texto não rotulado, resultando em avanços sem precedentes em uma vasta gama de tarefas de PLN, desde classificação de texto e resposta a perguntas até reconhecimento de entidades nomeadas.4 O BERT estabeleceu um novo paradigma: pré-treinar um modelo de linguagem massivo e, em seguida, ajustá-lo (fine-tuning) para tarefas específicas com conjuntos de dados menores.4

Este relatório tem como objetivo realizar uma análise técnica comparativa e aprofundada de uma seleção de modelos de linguagem proeminentes que emergiram na esteira do BERT, incluindo suas variantes diretas, otimizações e modelos com abordagens alternativas de pré-treinamento ou arquitetura. O foco reside nas metodologias de pré-treinamento, nas características arquitetônicas distintas, nos parâmetros de treinamento chave (como taxa de aprendizado, tamanho do lote e duração do treinamento) e na eficiência computacional associada. A análise baseia-se primordialmente em informações extraídas dos artigos de pesquisa originais e documentação técnica de fontes confiáveis, como o repositório Hugging Face.4 O propósito é fornecer um recurso consolidado e detalhado para pesquisadores, cientistas de dados e estudantes avançados na área de PLN, facilitando a compreensão das nuances e trade-offs entre esses influentes modelos.

Desde o advento do BERT, observa-se uma notável e rápida diversificação no desenvolvimento de modelos de linguagem. Pesquisadores buscaram não apenas otimizar o processo de treinamento do BERT original, que se mostrou computacionalmente caro e potencialmente sub-otimizado 13, como exemplificado pelo RoBERTa 13, mas também desenvolver modelos mais eficientes em termos de parâmetros e velocidade de inferência, como o DistilBERT 16 e o ALBERT.18 Além disso, foram propostos objetivos de pré-treinamento fundamentalmente distintos do Masked Language Modeling (MLM) e Next Sentence Prediction (NSP) do BERT, como o Replaced Token Detection (RTD) do ELECTRA 20 e posteriormente adotado pelo DeBERTaV3.22 Inovações arquitetônicas também surgiram, como a atenção desagregada do DeBERTa 23 e a incorporação de técnicas modernas de LLMs no ModernBERT.25 Essa proliferação indica que a otimização de modelos de linguagem pós-BERT seguiu múltiplos caminhos, explorando diferentes trade-offs entre performance, eficiência computacional, eficiência de dados e complexidade arquitetônica.

**2. Análise Comparativa de Modelos de Linguagem Fundacionais**

**2.1. BERT (Bidirectional Encoder Representations from Transformers)**

O BERT 4 marcou o início de uma nova era no PLN ao ser o primeiro modelo a aplicar com sucesso o pré-treinamento bidirecional profundo utilizando a arquitetura Transformer exclusivamente baseada em atenção. Sua capacidade de considerar simultaneamente o contexto à esquerda e à direita de uma palavra em todas as camadas permitiu a criação de representações contextuais mais ricas e eficazes do que as geradas por modelos unidirecionais (como LSTMs ou GPTs iniciais) ou por concatenações superficiais de representações esquerda-direita e direita-esquerda.5

* **Metodologia:** O pré-treinamento do BERT baseia-se em dois objetivos auto-supervisionados, aprendidos conjuntamente a partir de texto não rotulado:
  + **Masked Language Model (MLM):** Inspirado na tarefa Cloze 5, o MLM envolve mascarar aleatoriamente 15% dos tokens de entrada e treinar o modelo para prever o ID original desses tokens mascarados com base no contexto bidirecional.5 A estratégia de mascaramento específica é: 80% das vezes, o token é substituído pelo token especial ``; 10% das vezes, é substituído por um token aleatório do vocabulário; e 10% das vezes, o token original é mantido.28 Essa abordagem força o modelo a aprender representações contextuais robustas.
  + **Next Sentence Prediction (NSP):** Nesta tarefa binária, o modelo recebe dois segmentos de texto (A e B) e deve prever se o segmento B é a sentença que realmente sucede A no corpus original ou se é uma sentença aleatória.10 O objetivo declarado era capacitar o modelo a entender relações entre sentenças, crucial para tarefas como Question Answering (QA) e Natural Language Inference (NLI).10
* **Variantes Arquitetônicas:** O BERT foi lançado em duas configurações principais 19:
  + **BERT-Base:** Composto por 12 camadas (blocos Transformer), dimensão escondida (hidden size) de 768, 12 cabeças de atenção (attention heads) por camada, totalizando aproximadamente 110 milhões de parâmetros.19
  + **BERT-Large:** Uma versão maior com 24 camadas, dimensão escondida de 1024, 16 cabeças de atenção, e cerca de 334-340 milhões de parâmetros (pequenas variações são reportadas nas fontes 19).
* **Tokenização:** O BERT utiliza embeddings WordPiece com um vocabulário de 30.000 tokens.5 Foram disponibilizadas versões "uncased" (ignora capitalização) e "cased" (sensível à capitalização).28 A entrada para o modelo segue o formato Sequência A Sequência B para pares de sentenças, ou Sequência para uma única sentença, onde é um token especial usado para tarefas de classificação e delimita segmentos.32 A sequência máxima padrão é de 512 tokens.32
* **Parâmetros de Treinamento (Originais):**
  + Otimizador: Adam com β1=0.9, β2=0.999.5
  + Taxa de Aprendizado (LR): 1e-4.5
  + Agendamento de LR: Aquecimento (warmup) linear nos primeiros 10.000 passos, seguido por decaimento linear.5
  + Tamanho do Lote (Batch Size): 256 sequências.5
  + Passos/Épocas: 1.000.000 de passos. Com o corpus de pré-treinamento (Wikipedia + BookCorpus, ~3.3 bilhões de palavras), isso equivale a aproximadamente 40 épocas.5
  + Comprimento da Sequência: Treinado por 90% dos passos com sequências de 128 tokens e os 10% restantes com 512 tokens.28
  + Decaimento de Peso (Weight Decay): L2 de 0.01.5 Funcionalmente similar ao AdamW mencionado em outras fontes.14
* **Tempo/Hardware de Treinamento (Original):** O pré-treinamento original levou 4 dias.5
  + BERT-Base: 4 Cloud TPUs (total de 16 chips TPU).5
  + BERT-Large: 16 Cloud TPUs (total de 64 chips TPU).5
  + Posteriormente, otimizações como o otimizador LAMB demonstraram a possibilidade de reduzir drasticamente esse tempo em hardware mais poderoso (e.g., 76 minutos em 1024 chips TPUv3 14), mas o treinamento original foi um marco em termos de custo computacional na época.

O impacto do BERT foi imenso, estabelecendo o paradigma de pré-treinamento seguido de fine-tuning que dominou o PLN por vários anos.4 A demonstração da eficácia do pré-treinamento bidirecional profundo 5 foi sua contribuição central. Contudo, logo após seu lançamento, a comunidade de pesquisa identificou limitações. O estudo do RoBERTa 13 concluiu que o BERT original era significativamente "sub-treinado" (undertrained). Além disso, o custo computacional do pré-treinamento era substancial 14, motivando a busca por otimizações e modelos mais eficientes. A utilidade real da tarefa NSP também começou a ser questionada 15, abrindo caminho para abordagens alternativas.

**2.2. RoBERTa (A Robustly Optimized BERT Pretraining Approach)**

RoBERTa 13 surgiu como resultado de um estudo de replicação cuidadoso do pré-treinamento do BERT.13 Em vez de propor uma arquitetura radicalmente nova, RoBERTa focou em otimizar os hiperparâmetros de treinamento, a escala dos dados e refinar os objetivos de pré-treinamento, demonstrando que o BERT original estava, de fato, sub-treinado e que melhorias significativas poderiam ser alcançadas através de um regime de treinamento mais robusto.

* **Metodologia:** As principais modificações introduzidas pelo RoBERTa em relação ao BERT são 15:
  + **Remoção do NSP:** A tarefa de Next Sentence Prediction foi completamente removida. Os autores descobriram que treinar apenas com o objetivo MLM melhorava a performance em tarefas downstream, sugerindo que o NSP poderia ser prejudicial ou, no mínimo, desnecessário.15 O modelo é alimentado com blocos contíguos de texto, potencialmente de múltiplos documentos (formato FULL-SENTENCES).40
  + **Mascaramento Dinâmico:** Ao contrário do BERT, que aplicava uma máscara estática aos dados uma única vez durante o pré-processamento, RoBERTa gera um novo padrão de máscara para cada sequência de treinamento a cada época.36 Isso aumenta a diversidade dos dados de treinamento e melhora a robustez do modelo.
  + **Dados Maiores e Treinamento Mais Longo:** RoBERTa foi treinado com um volume de dados significativamente maior (160GB, incluindo BookCorpus, CC-News, OpenWebText e Stories) comparado aos 16GB do BERT.46 Além disso, o treinamento foi estendido por mais passos (500k vs 1M, mas com batch size muito maior) e/ou épocas.15
  + **Batches Maiores:** O tamanho do lote de treinamento foi drasticamente aumentado para 8.000 (8k) sequências.15 Isso, combinado com a otimização da taxa de aprendizado, foi crucial para a melhoria do desempenho.
  + **Sequências Longas:** O treinamento foi realizado exclusivamente com sequências de comprimento máximo (512 tokens) desde o início, eliminando a fase inicial de treinamento com sequências curtas (128 tokens) utilizada pelo BERT.39
* **Arquitetura:** A arquitetura do RoBERTa é essencialmente a mesma do BERT, com pequenas modificações nos embeddings.36
  + **RoBERTa-Large:** Possui 24 camadas, dimensão escondida de 1024, 16 cabeças de atenção e aproximadamente 355 milhões de parâmetros.44
* **Tokenização:** RoBERTa utiliza Byte-level Byte Pair Encoding (BPE), o mesmo método usado pelo GPT-2, com um vocabulário maior de 50.265 tokens.36 O formato de entrada é <s> Sequência A </s> para uma única sequência ou <s> Sequência A </s></s> Sequência B </s> para pares.36
* **Parâmetros de Treinamento (RoBERTa-Large):**
  + Otimizador: Adam com β1=0.9, β2=0.98, ε=1e-6.45
  + Taxa de Aprendizado (LR): Pico de 4e-4.45
  + Agendamento de LR: Aquecimento linear por 30.000 passos, seguido por decaimento linear.45
  + Tamanho do Lote (BS): 8.192 (8k) sequências.44
  + Passos: 500.000 passos.44
  + Comprimento da Sequência: 512 tokens.44
  + Decaimento de Peso (Weight Decay): 0.01.45
* **Tempo/Hardware de Treinamento:** O pré-treinamento do RoBERTa-Large levou aproximadamente 1 dia utilizando 1024 GPUs NVIDIA V100.39 Embora o tempo absoluto seja menor que os 4 dias do BERT-Large, a quantidade total de computação (GPU-horas) é consideravelmente maior devido ao número massivo de GPUs usadas em paralelo.

A principal contribuição do RoBERTa foi demonstrar empiricamente o quão crucial são a escala (dados, tamanho do lote, duração do treinamento) e a otimização cuidadosa dos hiperparâmetros e objetivos de pré-treinamento.13 As melhorias substanciais de desempenho sobre o BERT foram alcançadas sem alterações significativas na arquitetura Transformer subjacente.15 Isso estabeleceu um novo padrão para o pré-treinamento de modelos de linguagem grandes e destacou que ganhos significativos ainda poderiam ser obtidos refinando a receita de treinamento, um ponto que influenciou fortemente pesquisas subsequentes.

Adicionalmente, a decisão de remover a tarefa NSP 15 e o sucesso subsequente do RoBERTa solidificaram as dúvidas sobre a real contribuição do NSP para o aprendizado de representações linguísticas gerais. Isso incentivou a exploração de objetivos alternativos ou a confiança exclusiva no MLM (ou suas variantes) em modelos posteriores, como o ALBERT, que propôs o Sentence Order Prediction (SOP) como uma alternativa mais significativa para capturar coerência inter-sentencial 18, e o ELECTRA, que abandonou completamente tarefas explícitas de nível sentencial em favor da discriminação em nível de token.20

**2.3. DistilBERT**

DistilBERT 16 representa uma abordagem focada na eficiência, buscando criar uma versão menor, mais rápida e mais leve do BERT, adequada para cenários com restrições computacionais, como dispositivos móveis (edge devices) ou aplicações que exigem baixa latência, sem sacrificar drasticamente a performance.16

* **Metodologia:** A técnica central empregada pelo DistilBERT é a **Knowledge Distillation** (Destilação de Conhecimento) 17, aplicada durante a fase de pré-treinamento.
  + Nesse processo, um modelo menor ("estudante", DistilBERT) é treinado para imitar o comportamento de um modelo maior e já treinado ("professor", neste caso, BERT-Base).56 A imitação não se baseia apenas nas previsões "duras" (a classe final prevista), mas nas probabilidades "suaves" (logits ou probabilidades pós-softmax) geradas pelo professor.17 Essas probabilidades suaves contêm informações mais ricas sobre as relações que o professor aprendeu entre as classes.
  + DistilBERT utiliza uma **triple loss** (função de perda tripla) para guiar o aprendizado do estudante 16:
    1. **Distillation Loss (L\_ce):** Mede a divergência (geralmente Kullback-Leibler - KL divergence) entre as distribuições de probabilidade suavizadas (usando um parâmetro de temperatura T > 1) do estudante e do professor sobre o vocabulário.17 A temperatura suaviza as probabilidades, permitindo que o estudante aprenda com as "incertezas" do professor.17
    2. **Masked Language Modeling Loss (L\_MLM):** Uma perda padrão de MLM (cross-entropy), similar à do BERT, treinando o estudante para prever tokens mascarados diretamente dos dados.17 DistilBERT não utiliza o objetivo NSP.55
    3. **Cosine Embedding Loss (L\_cos):** Maximiza a similaridade de cosseno entre os vetores de hidden states do estudante e do professor.16 Isso força o estudante a aprender não apenas as previsões de saída, mas também as representações internas do professor.
* **Arquitetura:** DistilBERT utiliza a mesma arquitetura Transformer do BERT, mas reduz o número de camadas pela metade. A versão padrão (distilbert-base-uncased) tem 6 camadas, em comparação com as 12 do bert-base-uncased.56 A inicialização dos pesos do estudante é feita copiando-se camadas alternadas do professor BERT-Base.65 Essa redução de camadas resulta em um modelo com 66 milhões de parâmetros, 40% menor que o BERT-Base (110M).16
* **Tokenização:** DistilBERT herda o tokenizer do seu modelo professor. Por exemplo, distilbert-base-uncased usa o mesmo tokenizer WordPiece (uncased) do bert-base-uncased.33 Uma diferença notável é que DistilBERT não utiliza token\_type\_ids (segment embeddings), simplificando a entrada.33
* **Parâmetros de Treinamento:**
  + Otimizador: Adam [135 (implícito)].
  + Taxa de Aprendizado (LR): Os snippets não especificam o LR de pré-treinamento.68 Para fine-tuning, valores como 2e-5 a 5e-5 são recomendados.63
  + Tamanho do Lote (BS): Treinado com lotes grandes usando acumulação de gradiente (até 4.000 exemplos por lote efetivo).64
  + Passos/Épocas: Não especificado nos snippets.68
  + Outros: Utiliza mascaramento dinâmico, como RoBERTa.64 A temperatura T para a distillation loss é > 1.17
* **Tempo/Hardware de Treinamento:** O pré-treinamento do DistilBERT levou aproximadamente 90 horas em 8 GPUs NVIDIA V100 (16GB).51 Isso representa uma redução significativa no custo computacional em comparação com o treinamento do BERT-Base ou RoBERTa-Base.16 Em termos de inferência, DistilBERT é reportado como sendo 60% mais rápido que BERT-Base.16

DistilBERT foi um marco ao demonstrar a viabilidade da destilação de conhecimento durante o pré-treinamento para criar modelos de linguagem de propósito geral eficientes.16 Antes dele, a destilação era mais comumente aplicada para criar modelos específicos para tarefas após o fine-tuning. Ao integrar a destilação na fase de pré-treinamento, DistilBERT ofereceu um modelo que poderia ser usado como um substituto mais leve e rápido para o BERT-Base em muitas aplicações, especialmente aquelas com restrições de recursos ou latência.16 A capacidade de reter cerca de 97% do desempenho do BERT no benchmark GLUE com 40% menos parâmetros e 60% mais velocidade 16 validou a eficácia da abordagem e ofereceu um trade-off atraente para a comunidade.

A função de perda tripla 16 foi um componente chave para o sucesso dessa transferência de conhecimento. A distillation loss (L\_ce) permitiu ao estudante aprender a distribuição de saída "suavizada" do professor, capturando nuances além da previsão correta. A loss de MLM (L\_mlm) garantiu que o estudante ainda aprendesse a modelar a linguagem diretamente dos dados brutos. Finalmente, a cosine embedding loss (L\_cos) forçou um alinhamento das representações internas (hidden states), transferindo conhecimento estrutural mais profundo do professor para o estudante. Essa combinação multifacetada de sinais de treinamento provavelmente foi fundamental para minimizar a perda de desempenho apesar da compressão significativa do modelo.

**2.4. ALBERT (A Lite BERT for Self-supervised Learning)**

ALBERT 1 foi proposto com o objetivo principal de construir modelos de linguagem pré-treinados mais escaláveis, abordando as limitações de memória e o tempo de treinamento crescente associados ao aumento do tamanho dos modelos tipo BERT.18 Para isso, introduziu técnicas inovadoras de redução de parâmetros.

* **Metodologia:** As principais contribuições metodológicas do ALBERT são 18:
  + **Factorized Embedding Parameterization:** Em modelos como BERT, a dimensão dos embeddings de vocabulário (E) é diretamente ligada à dimensão da camada escondida (H). ALBERT desacopla essas dimensões, primeiro projetando os tokens one-hot em um espaço de embedding de baixa dimensão (E, tipicamente 128) e, em seguida, projetando esses embeddings de baixa dimensão para o espaço da camada escondida (H).18 Como geralmente H >> E, isso reduz drasticamente o número de parâmetros na camada de embedding (de V x H para V x E + E x H, onde V é o tamanho do vocabulário). A redução é mais significativa para modelos com H grande e V grande.
  + **Cross-Layer Parameter Sharing:** Diferentemente do BERT, onde cada camada Transformer tem seus próprios pesos, ALBERT compartilha todos os parâmetros (tanto das redes feed-forward quanto dos mecanismos de atenção) entre todas as camadas.18 Isso impede que o número de parâmetros cresça linearmente com a profundidade do modelo. Essa técnica também atua como uma forma de regularização, o que pode estabilizar o treinamento e melhorar a generalização.19
  + **Sentence Order Prediction (SOP):** ALBERT substitui a tarefa NSP do BERT pela SOP.18 Na SOP, o modelo recebe dois segmentos de texto que são sempre consecutivos no corpus original e deve prever se eles estão na ordem correta ou se foram trocados. Os autores argumentam que NSP é uma tarefa muito fácil (mistura predição de tópico com coerência) e que SOP foca mais diretamente na modelagem da coerência e coesão inter-sentencial.18
* **Variantes Arquitetônicas:** ALBERT foi lançado em várias configurações, incluindo base, large, xlarge e xxlarge [18 (Table), 19 (Table)]. A versão xxlarge é a maior.
  + **ALBERT-xxlarge:** A configuração exata varia ligeiramente entre v1 e v2. A arquitetura base tem 12 blocos Transformer únicos que são repetidos N vezes (N não é explicitamente definido, mas o número total de camadas efetivas é alto). Possui H=4096, E=128, A=64 cabeças de atenção. O número de parâmetros é de 235M para v1 e 223M para v2.18 Notavelmente, ALBERT-xxlarge v2 tem menos parâmetros que BERT-Large (223M vs ~340M), apesar de ter uma dimensão escondida muito maior (4096 vs 1024). A versão v2 difere da v1 por ter taxas de dropout diferentes, dados de treinamento adicionais e treinamento mais longo.79
* **Tokenização:** Utiliza SentencePiece com um vocabulário de 30.000 tokens, e é uncased.72
* **Parâmetros de Treinamento (ALBERT-xxlarge v2):** Detalhes específicos do pré-treinamento para a versão xxlarge v2 não estão disponíveis nos materiais fornecidos.79 No entanto, informações gerais e de outras versões sugerem:
  + Otimizador: LAMB foi usado para treinar modelos ALBETO (versão espanhola) 72 e também em experimentos com ALBERT-Base.84 O paper original menciona Adam com learning rate de 1e-4 para BERT-large, mas não especifica para ALBERT.
  + Taxa de Aprendizado (LR): Não especificada para pré-treinamento.79 Para fine-tuning da v2 na tarefa RACE, foi usado 1e-5.85
  + Tamanho do Lote (BS): O pré-treinamento original do ALBERT usou um batch size global de 4096.84
  + Passos/Épocas: Não especificado.79 ALBERT-base foi treinado por 1M passos.87 A versão v2 foi treinada por mais tempo que a v1.79
  + Comprimento da Sequência: Máximo de 512 tokens.76
* **Tempo/Hardware de Treinamento:** Não especificado para pré-treinamento do xxlarge-v2.79 O pré-treinamento do ALBERT-base levou 8 dias em uma Cloud TPU V3.87 Em geral, ALBERT treina cerca de 1.7x mais rápido por época que um BERT de configuração similar (mesmo número de camadas e H) devido à redução de parâmetros e comunicação.19 O fine-tuning do xxlarge v2 em 64 GPUs V100 levou 48 minutos para 2 épocas na tarefa SQuAD 2.0.88 O pré-treinamento de ALBERT-base/large em 8 nós p3dn.24xlarge levou 20/59 horas, respectivamente.86

A introdução do ALBERT destacou uma importante distinção entre eficiência de parâmetros (redução do número de parâmetros e, consequentemente, do tamanho do modelo em memória) e eficiência computacional (velocidade de treinamento ou inferência).18 As técnicas de fatorização de embedding e compartilhamento de parâmetros entre camadas reduziram drasticamente a contagem de parâmetros – ALBERT-large tem apenas 18M de parâmetros, comparado aos 334M do BERT-large.19 Isso torna os modelos ALBERT significativamente mais leves em termos de armazenamento e memória RAM necessária. No entanto, como as camadas compartilhadas ainda precisam ser computadas sequencialmente durante o passo forward, o custo computacional por passo de treinamento ou inferência não diminui na mesma proporção.74 Em alguns casos, devido à maior profundidade efetiva ou maior dimensão escondida (como no xxlarge), a inferência pode até ser mais lenta que a de um BERT de tamanho comparável em parâmetros.53 ALBERT, portanto, oferece um trade-off específico: memória muito reduzida e potencial de regularização aprimorado, mas sem uma aceleração computacional proporcional.

A substituição do NSP pelo SOP 18 também foi uma contribuição notável. Seguindo o questionamento do RoBERTa sobre a utilidade do NSP, ALBERT propôs SOP como uma tarefa de pré-treinamento inter-sentencial mais desafiadora e focada na coerência discursiva. Isso reflete a busca contínua na comunidade por objetivos de auto-supervisão mais eficazes que capturem diferentes aspectos da compreensão da linguagem, indo além da simples previsão de palavras mascaradas.

**2.5. ELECTRA (Efficiently Learning an Encoder that Classifies Token Replacements Accurately)**

ELECTRA 20 introduziu uma abordagem de pré-treinamento radicalmente diferente do MLM, visando maior eficiência de amostra e computacional.20

* **Metodologia:** A inovação central do ELECTRA é a tarefa de **Replaced Token Detection (RTD)**, implementada através de uma arquitetura com dois componentes 2:
  + **Generator (G):** Um modelo de linguagem menor (geralmente um MLM como BERT, tipicamente com 1/4 a 1/2 do tamanho do Discriminator) que recebe uma sequência de entrada com alguns tokens mascarados (e.g., 15%) e preenche essas máscaras com tokens amostrados de sua distribuição de saída.20 O objetivo do gerador é produzir substituições plausíveis, mas não necessariamente corretas.
  + **Discriminator (D):** O modelo principal, que recebe a sequência corrompida pelo gerador (com os tokens originais e os substituídos) e é treinado para prever, para *cada* token na sequência, se ele é um token original ("real") ou se foi substituído pelo gerador ("fake").20 Esta é a tarefa RTD.
  + **Eficiência de Amostra:** A principal vantagem alegada é a eficiência de amostra. Enquanto o MLM calcula a perda apenas sobre os tokens mascarados (~15% da entrada), a tarefa RTD do discriminador calcula a perda sobre *todos* os tokens da sequência.20 Isso significa que o modelo aprende um sinal de supervisão muito mais denso a cada exemplo, tornando o pré-treinamento mais eficiente em termos de dados e computação.
  + **Pós-Treinamento:** Após o pré-treinamento, o gerador é descartado, e apenas o discriminador é utilizado para fine-tuning em tarefas downstream.91
  + **Compartilhamento de Embeddings:** No design original, os embeddings de token são compartilhados entre o gerador e o discriminador [101 (implícito), 99].
* **Arquitetura:** A arquitetura do discriminador ELECTRA é baseada no Transformer, muito similar ao BERT.101 Uma modificação opcional permite que o embedding\_size seja menor que o hidden\_size, com uma camada de projeção linear entre eles para eficiência.101
  + **ELECTRA-Large (Discriminator):** Possui 24 camadas, dimensão escondida de 1024, 16 cabeças de atenção e 335 milhões de parâmetros.103
* **Tokenização:** Geralmente utiliza o tokenizer WordPiece herdado do BERT, como o vocabulário bert-base-uncased.96
* **Parâmetros de Treinamento (ELECTRA-Large Discriminator):** Os snippets fornecidos não contêm os detalhes específicos do pré-treinamento do ELECTRA-Large.96 No entanto, informações do paper original (Clark et al., 2020) e de trabalhos relacionados indicam:
  + Otimizador: Adam [105 (implícito), 136].
  + Taxa de Aprendizado (LR): 5e-4 (para Large, segundo paper original).
  + Tamanho do Lote (BS): 2048 (para Large, segundo paper original).
  + Passos: 400.000 passos (para Large, segundo paper original).
  + Comprimento da Sequência: 512 (segundo paper original).
  + Outros: O peso da loss do discriminador na loss combinada (Generator MLM + Discriminator RTD) é tipicamente alto, λ=50.104 O tamanho do gerador é um hiperparâmetro importante, geralmente menor que o discriminador.
* **Tempo/Hardware de Treinamento:** ELECTRA demonstra eficiência computacional notável. ELECTRA-Large alcança performance comparável a RoBERTa-Large e XLNet usando menos de 1/4 do custo computacional de pré-treinamento.21 ELECTRA-Small pode ser treinado em cerca de 4 dias em uma única GPU V100 21, um feito impressionante para a época.

A inovação fundamental do ELECTRA reside na tarefa de pré-treinamento RTD, que se mostrou significativamente mais eficiente em termos de amostra do que o MLM.20 Ao exigir que o discriminador faça uma previsão para cada token de entrada (original vs. substituído), em vez de apenas para os 15% mascarados, ELECTRA extrai um sinal de aprendizado muito mais rico de cada exemplo de treinamento. Essa maior densidade de sinal permite que modelos ELECTRA alcancem níveis de performance comparáveis aos de modelos baseados em MLM com substancialmente menos computação de pré-treinamento, ou superem modelos de tamanho similar treinados com MLM pelo mesmo custo computacional.20 Isso foi particularmente impactante para modelos menores (ELECTRA-Small), que demonstraram performance surpreendentemente forte.21

No entanto, a dinâmica entre o gerador e o discriminador introduz suas próprias complexidades. O desempenho do discriminador (o modelo final) é sensível ao tamanho e à capacidade do gerador.91 Se o gerador for muito fraco, a tarefa RTD se torna trivial e o discriminador aprende pouco. Se o gerador for muito forte (ou seja, suas substituições forem muito boas), ele pode dificultar excessivamente a tarefa para o discriminador ou até mesmo prejudicar seu aprendizado. Além disso, o compartilhamento de embeddings de token entre o gerador (treinado com MLM, que favorece a proximidade de embeddings semanticamente similares) e o discriminador (treinado com RTD, que favorece a distinção entre tokens, mesmo que similares) cria uma tensão inerente, descrita como "cabo de guerra" (tug-of-war).22 Essa dinâmica requer um balanceamento cuidadoso (e.g., através do peso da loss λ e do tamanho relativo G/D) e motivou pesquisas posteriores, como a técnica GDES no DeBERTaV3, para mitigar esses efeitos.

**2.6. DeBERTa (Decoding-enhanced BERT with Disentangled Attention) & DeBERTaV3**

DeBERTa 3 e sua evolução, DeBERTaV3 12, representam avanços significativos tanto na arquitetura quanto na metodologia de pré-treinamento em relação ao BERT e RoBERTa.

* **Metodologia (DeBERTa V1/V2):** As inovações chave do DeBERTa original (V1/V2) focaram em melhorar a forma como as informações de conteúdo e posição são representadas e utilizadas 23:
  + **Disentangled Attention:** Ao contrário do BERT que soma os embeddings de conteúdo e posição, DeBERTa representa cada token por dois vetores separados: um para o conteúdo e outro para a posição (relativa).23 O cálculo da atenção é então desagregado em quatro componentes (conteúdo-conteúdo, conteúdo-posição, posição-conteúdo, posição-posição - embora a última seja frequentemente omitida), usando matrizes de transformação distintas. Argumenta-se que isso modela de forma mais eficaz a importância relativa do conteúdo e da posição nas relações entre palavras.
  + **Enhanced Mask Decoder (EMD):** Enquanto a atenção desagregada foca nas posições relativas, o EMD reintroduz informações de posição absoluta na camada final de predição do MLM.23 Isso reconhece que a posição absoluta de uma palavra também pode ser importante para prever tokens mascarados.
  + **Virtual Adversarial Training (SiFT):** Aplicado durante o fine-tuning, o SiFT (Scale-invariant Fine-Tuning) introduz perturbações adversariais normalizadas nos embeddings para melhorar a robustez e generalização do modelo em tarefas downstream.24
* **Metodologia (DeBERTaV3):** DeBERTaV3 12 baseia-se na arquitetura DeBERTa, mas substitui o objetivo de pré-treinamento MLM pelo RTD (Replaced Token Detection), inspirado no ELECTRA, para capitalizar sua maior eficiência de amostra.22 A principal inovação adicional é:
  + **Gradient-Disentangled Embedding Sharing (GDES):** Para resolver o problema do "cabo de guerra" (tug-of-war) identificado no compartilhamento de embeddings do ELECTRA, GDES modifica o processo de backpropagation.12 Essencialmente, os gradientes da loss do gerador (MLM) não atualizam os embeddings usados pelo discriminador, e os gradientes da loss do discriminador (RTD) não atualizam os embeddings usados pelo gerador, embora ambos possam derivar de uma matriz de embedding subjacente compartilhada. Isso isola as atualizações e melhora a estabilidade e a performance.
* **Arquitetura:** Baseada na arquitetura Transformer com as modificações da atenção desagregada.
  + **DeBERTa-Large (V1/V2):** Tipicamente 24 camadas, hidden size 1024, 16 cabeças de atenção, ~350-400M parâmetros. Uma versão V2 com 48 camadas e 1.5 bilhão de parâmetros também foi treinada, alcançando SOTA no SuperGLUE.23
  + **DeBERTa-V3-Large:** 24 camadas, hidden size 1024, 16 cabeças de atenção. Possui 304M parâmetros no backbone Transformer, mas devido ao vocabulário maior (128k), a camada de embedding adiciona 131M, totalizando cerca de 435M parâmetros.12
* **Tokenização:**
  + DeBERTa V1/V2: Dependia do modelo base (e.g., RoBERTa usava BPE).
  + DeBERTa V3: Utiliza um tokenizer SentencePiece com um vocabulário expandido de 128.000 tokens.12
* **Parâmetros de Treinamento (DeBERTa-V3-Large):** Os snippets fornecidos não detalham os parâmetros de pré-treinamento.12
  + Otimizador: Provavelmente AdamW (padrão para Transformers).
  + Taxa de Aprendizado (LR): Não especificada.12 LR de 3e-5 é usado para fine-tuning no MNLI.118
  + Tamanho do Lote (BS): Não especificado.12
  + Passos/Épocas: Não especificado.12
  + Comprimento da Sequência: Máximo padrão de 512, mas a atenção relativa permite teoricamente sequências mais longas.119
  + Dados: Treinado com o mesmo conjunto de dados de 160GB do RoBERTa e DeBERTa V2.12
* **Tempo/Hardware de Treinamento:** Não especificado nos snippets.12 O pré-treinamento de modelos desta escala é conhecido por ser computacionalmente intensivo.106

A introdução da atenção desagregada no DeBERTa 23 foi uma inovação arquitetônica chave. Ao tratar conteúdo e posição como fontes distintas de informação e modelar suas interações de forma mais explícita, DeBERTa conseguiu capturar relações contextuais de forma mais eficaz do que os modelos anteriores que simplesmente somavam os embeddings. Isso se refletiu em melhorias de desempenho, mesmo quando treinado com menos dados que o RoBERTa 23, indicando que o design arquitetônico em si oferecia vantagens intrínsecas e que havia espaço para refinar o próprio mecanismo de atenção do Transformer.

DeBERTaV3 12 exemplifica uma tendência de síntese no desenvolvimento de modelos. Ele combina a arquitetura avançada do DeBERTa (atenção desagregada) com a metodologia de pré-treinamento eficiente em amostra do ELECTRA (RTD). Reconhecendo o problema do "cabo de guerra" no compartilhamento de embeddings do ELECTRA, os autores desenvolveram o GDES 22 como uma solução específica para permitir que essas duas abordagens bem-sucedidas coexistissem harmoniosamente. Isso demonstra uma abordagem madura para o desenvolvimento de modelos, onde as melhores características de diferentes linhagens de pesquisa são identificadas, combinadas e os conflitos resultantes são resolvidos através de novas inovações técnicas, levando a um novo estado da arte.12

**2.7. alBERTina**

alBERTina 3 é uma família de modelos de linguagem fundacionais desenvolvida especificamente para a língua portuguesa, com versões distintas para as variantes europeia (PT-PT) e brasileira (PT-BR).

* **Metodologia:** O ponto crucial é que, apesar do nome sugerir uma ligação com o modelo ALBERT, alBERTina é, na verdade, **baseado na arquitetura DeBERTa**.3 O nome é uma homenagem e não reflete a arquitetura subjacente. A metodologia envolveu o pré-treinamento (continuado ou do zero) desses modelos DeBERTa em grandes corpora de texto em português.
  + A versão alBERTina-1.5B-PTPT, por exemplo, usou o modelo microsoft/deberta-v2-xxlarge (um modelo inglês de 1.5B parâmetros) como ponto de partida.124
  + Os corpora de treinamento variam: a versão PT-PT utilizou uma compilação de fontes abertas como CulturaX, OSCAR e Europarl.3 A versão PT-BR original usou o corpus brWaC 113, enquanto versões posteriores (com licença mais permissiva) usaram outros corpora.112
* **Variantes Arquitetônicas:** A família alBERTina inclui modelos de diferentes tamanhos para ambas as variantes do português, tipicamente 100 milhões, 900 milhões e 1.5 bilhão de parâmetros.3
  + **alBERTina-1.5B-PTPT:** Baseado no DeBERTa V2 xxlarge, possui 48 camadas e hidden size de 1536.124
* **Tokenização:** Utiliza o tokenizer SentencePiece do modelo DeBERTa base (V2 ou V3), que possui um vocabulário grande de 128.000 tokens [107 (ViDeBERTa), 3].
* **Parâmetros de Treinamento:** Os parâmetros variam ligeiramente entre as versões. Para as versões PT-PT:
  + Otimizador: Provavelmente AdamW (padrão).
  + Taxa de Aprendizado (LR): 1e-5 com decaimento linear e 10k passos de warmup (para a versão 900M).3
  + Tamanho do Lote (BS): Efetivo de 832 (900M, com acumulação de gradiente) ou 896 (1.5B, sem acumulação).3
  + Passos/Épocas: Treinamento definido por número de passos ou tempo. A versão 1.5B foi treinada por 250k passos (seq 128) + 80k passos (seq 256) + 60k passos (seq 512).124
  + Comprimento da Sequência: Estratégia de treinamento progressivo, aumentando o comprimento da sequência (128 -> 256 -> 512) durante o treinamento.124
* **Tempo/Hardware de Treinamento:**
  + alBERTina 900M PT-PT: 96 horas em 32 GPUs V100 (16GB).3
  + alBERTina 1.5B PT-PT: Total de 96 horas (48h + 24h + 24h) em um nó Google Cloud A2 com 16 GPUs A100 (40GB).124

O desenvolvimento do alBERTina 3 sublinha a importância e a eficácia da criação de modelos de linguagem monolingues de alta qualidade para línguas além do inglês. Embora modelos multilingues como mBERT e XLM-RoBERTa 113 ofereçam cobertura para muitas línguas, incluindo o português, modelos treinados especificamente em grandes corpora da língua alvo tendem a apresentar desempenho superior em tarefas nessa língua [113 (comparação implícita com BERTimbau), 65]. A existência prévia do BERTimbau 113 para PT-BR já havia indicado essa vantagem sobre o mBERT. alBERTina, ao alcançar o estado da arte para o português 3, reforça a validade dessa abordagem focada na língua.

Além disso, a escolha do DeBERTa 3 como arquitetura base para o alBERTina ilustra uma estratégia pragmática e eficaz para o desenvolvimento de modelos para línguas com menos recursos (em termos de pesquisa e investimento direto) do que o inglês. Ao invés de desenvolver uma nova arquitetura do zero, a equipe adaptou uma arquitetura de ponta (DeBERTa V2 xxlarge 124), que já havia demonstrado excelente desempenho em inglês, e a (re)treinou extensivamente em dados portugueses.3 Isso permite transferir os avanços arquitetônicos (como a atenção desagregada) para a nova língua, focando o esforço na coleta e curadoria de dados e no processo de treinamento específico para o idioma.

**2.8. ModernBERT**

ModernBERT 25 é apresentado como uma "modernização" da arquitetura BERT, com o objetivo explícito de incorporar muitas das inovações desenvolvidas na era dos Large Language Models (LLMs) baseados em decoders de volta aos modelos encoder, visando melhor desempenho, eficiência e capacidade de lidar com contextos mais longos.25

* **Metodologia:** As principais inovações metodológicas e arquitetônicas incluem:
  + **Rotary Positional Embeddings (RoPE):** Em vez de embeddings posicionais absolutos ou relativos tradicionais, ModernBERT usa RoPE.25 RoPE aplica rotações em subespaços dos embeddings de query e key com base na posição, permitindo que a atenção capture informações de posição relativa de forma mais flexível e escalável, suportando nativamente sequências de até 8192 tokens.25
  + **Alternating Attention:** Para lidar eficientemente com sequências longas, ModernBERT alterna entre camadas de atenção global (onde cada token atende a todos os outros) e camadas de atenção local (onde cada token atende apenas a uma janela deslizante de tokens vizinhos, e.g., 128).25 A atenção global ocorre a cada N camadas (e.g., N=3 25). Diferentes parâmetros RoPE (theta) são usados para atenção global e local.25
  + **GeGLU Activation:** A função de ativação GeLU, comum em BERTs, é substituída pela GeGLU (Gated GELU Linear Unit), que demonstrou melhor desempenho em LLMs.25
  + **Normalization and Architecture:** Utiliza pré-normalização (LayerNorm antes dos blocos de atenção e FFN) para maior estabilidade de treinamento, com uma LayerNorm adicional após os embeddings.125 Remove bias desnecessários na arquitetura.127
  + **Efficiency Optimizations:** Incorpora técnicas como "unpadding" (ignora tokens de padding no cálculo) e suporte para Flash Attention 2, que otimizam o uso de memória e a velocidade de computação, especialmente para sequências longas.25
  + **Training Data:** Pré-treinado em uma escala massiva de 2 trilhões de tokens, incluindo documentos da web, código e literatura científica, com menos repetições de dados do que era comum em encoders anteriores.25
* **Variantes Arquitetônicas:** Disponível nas versões Base e Large.26
  + **ModernBERT-Base:** 22 camadas, hidden size 768, 12 cabeças de atenção, 149 milhões de parâmetros.25
  + **ModernBERT-Large:** 28 camadas, hidden size 1024 (provável), 16 cabeças de atenção (provável), 395 milhões de parâmetros.25
* **Tokenização:** Utiliza um vocabulário de 50.368 tokens.25 Importante notar que ModernBERT **não** utiliza token\_type\_ids (segment embeddings).26
* **Parâmetros de Treinamento:**
  + Otimizador: StableAdamW.129
  + Taxa de Aprendizado (LR): Agendamento trapezoidal com decaimento 1-sqrt.130 O valor exato não é fornecido.
  + Tamanho do Lote (BS): Não especificado.
  + Passos/Épocas: Não especificado, mas treinado sobre 2 trilhões de tokens.25
  + Comprimento da Sequência: Pré-treinado inicialmente com sequências de até 1024 tokens, depois estendido para 8192 tokens.130
* **Tempo/Hardware de Treinamento:** Pré-treinado em 8 GPUs NVIDIA H100.130 A duração total do treinamento não foi especificada.

ModernBERT exemplifica um esforço deliberado de "backporting", ou seja, aplicar inovações desenvolvidas primariamente no contexto de LLMs generativos (decoders) de volta aos modelos encoder.26 Técnicas como RoPE, GeGLU, otimizações como Flash Attention, e o treinamento em escala massiva (2T tokens) são características que se tornaram proeminentes com LLMs recentes e foram integradas ao ModernBERT. Isso demonstra um reconhecimento de que os avanços em diferentes famílias de arquiteturas Transformer podem ser sinérgicos e que os encoders, que permanecem cruciais para muitas tarefas discriminativas e de representação 26, podem se beneficiar dessas modernizações.

Um foco central do ModernBERT é superar duas limitações históricas dos encoders tipo BERT: a restrição de comprimento de contexto (tipicamente 512 tokens 32) e a eficiência computacional, especialmente com sequências mais longas. Ao adotar RoPE e atenção alternada (local/global), ModernBERT estende nativamente o comprimento de contexto para 8192 tokens.25 Combinado com otimizações como unpadding e Flash Attention 2 25, isso não apenas permite o processamento de documentos longos ou código, mas também o faz de maneira eficiente em termos de velocidade e memória.126 Essa combinação de longo contexto e eficiência abre novas possibilidades para aplicações de encoders em domínios anteriormente desafiadores.

**2.9. Humano (Baseline Humano)**

Incluir um "modelo" humano nesta comparação serve como um ponto de referência fundamental, embora os conceitos de treinamento e parâmetros de modelos de machine learning (ML) não se apliquem diretamente.

* **Metodologia ("Treinamento"):** A aquisição da linguagem humana é um processo biológico e cognitivo complexo que ocorre ao longo de muitos anos. Envolve exposição massiva a dados linguísticos (fala, texto) e não linguísticos (contexto visual, social), interação social, e o desenvolvimento de estruturas neurais dedicadas no cérebro. Não é um processo de otimização de parâmetros via gradiente descendente sobre uma função de perda definida [Query point (9)]. A comparação com o desempenho humano em benchmarks é comum.31
* **Tokenização:** Humanos processam a linguagem em múltiplos níveis de abstração simultaneamente – fonemas, morfemas, palavras, sintaxe, semântica, pragmática – de forma flexível e dependente do contexto. Não há um algoritmo fixo de subpalavras como WordPiece ou BPE.
* **Parâmetros de Treinamento:** Conceitos como taxa de aprendizado, tamanho do lote ou número de épocas não têm análogos diretos na aquisição de linguagem humana [Query point (9)]. A "escala" do aprendizado é a vida inteira de exposição e uso da linguagem.
* **Tempo/Hardware de Treinamento:** O processo leva anos, desde a infância até a idade adulta, para atingir proficiência completa (~20+ anos). O "hardware" é o cérebro humano, um sistema computacional de natureza biológica radicalmente diferente das GPUs/TPUs.
* **Papel como Benchmark:** O desempenho humano em tarefas de PLN bem definidas (como as encontradas nos benchmarks GLUE, SuperGLUE, SQuAD) é frequentemente usado como uma linha de base ou um "teto" para avaliar o progresso dos modelos de IA.23 O objetivo de muitos modelos é alcançar ou superar o desempenho humano nessas tarefas específicas. Notavelmente, modelos como o DeBERTa já superaram a média humana em benchmarks agregados como o SuperGLUE.23

Embora modelos de IA tenham superado o desempenho humano em benchmarks específicos e bem definidos, como o SuperGLUE 23, é crucial entender as limitações dessa comparação. O desempenho humano nesses benchmarks representa a capacidade em tarefas restritas e artificiais, não a totalidade da compreensão linguística humana. A capacidade humana de generalização para novas situações, o raciocínio de senso comum, a compreensão profunda do mundo, a criatividade linguística, a eficiência no aprendizado (aprendendo com poucos exemplos ou instruções explícitas) e a habilidade de usar a linguagem para interação social complexa ainda superam vastamente as capacidades dos modelos de linguagem atuais. Portanto, enquanto o desempenho humano serve como um marco útil, alcançar ou exceder esse marco em um benchmark específico não equivale a alcançar inteligência linguística de nível humano ou Inteligência Artificial Geral (AGI). O "alvo" da inteligência humana continua sendo muito mais amplo e complexo do que o medido pelos benchmarks atuais.

**3. Tabela Comparativa Resumida**

A tabela a seguir consolida as informações chave sobre os modelos analisados, permitindo uma comparação direta de suas metodologias, arquiteturas e parâmetros de pré-treinamento originais. Onde a informação não estava explicitamente disponível nos materiais de pesquisa fornecidos, indica-se "Não especificado".

| **Modelo** | **Metodologia (Resumo)** | **Tokenização** | **Taxa de Aprendizado (Pico)** | **Tamanho do Lote (Global)** | **Passos/Épocas de Pré-treinamento** | **Outros Parâmetros (L, H, A, E, Params)** | **Tempo/Hardware de Pré-treinamento Original** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **BERT-Base** | MLM + NSP; Bidirecionalidade profunda. 28 | WordPiece (30k) | 1e-4 | 256 | 1M Passos (~40 Épocas) 5 | L=12, H=768, A=12, E=768, P=110M 19 | 4 dias / 4 Cloud TPUs (16 chips) 5 |
| **BERT-Large** | MLM + NSP; Bidirecionalidade profunda. 28 | WordPiece (30k) | 1e-4 | 256 | 1M Passos (~40 Épocas) 5 | L=24, H=1024, A=16, E=1024, P=~340M 19 | 4 dias / 16 Cloud TPUs (64 chips) 5 |
| **RoBERTa-Large** | MLM Otimizado (sem NSP, mascaramento dinâmico, dados/batches/passos maiores). 15 | BPE nível de byte (50k) | 4e-4 | 8192 (8k) | 500k Passos 45 | L=24, H=1024, A=16, E=1024, P=355M 45 | ~1 dia / 1024 GPUs V100 39 |
| **DistilBERT (Base)** | Destilação de Conhecimento (do BERT-Base) + MLM + Cosine Loss; Sem NSP. 16 | WordPiece (30k, herdado) | Não especificado 69 | Até 4096 (c/ acumulação) | Não especificado 69 | L=6, H=768, A=12, E=768, P=66M 51 | ~90 horas / 8 GPUs V100 (16GB) 51 |
| **ALBERT-xxlarge-v2** | MLM + SOP; Fatorização de Embedding; Compartilhamento de Parâmetros entre Camadas. 18 | SentencePiece (30k) | Não especificado 79 | 4096 (provável) 84 | Não especificado 79 | L=12 (repetido), H=4096, A=64, E=128, P=223M 19 | Não especificado para xxlarge-v2.79 Large levou 59h/8 nós p3dn.86 |
| **ELECTRA-Large** | Generator (MLM) + Discriminator (RTD); Foco na eficiência de amostra. 20 | WordPiece (30k, herdado) | 5e-4 (paper original) | 2048 (paper original) | 400k Passos (paper original) | L=24, H=1024, A=16, E=1024 (ou <H), P=335M 102 | <1/4 compute de RoBERTa.21 Small treinou 4 dias/1 V100.97 |
| **DeBERTa-Large (V1/V2)** | MLM + Atenção Desagregada + Enhanced Mask Decoder. 23 | WordPiece/BPE | Não especificado | Não especificado | Não especificado | L=24, H=1024, A=16, E=1024, P=~350-400M (1.5B com L=48) 23 | Treinado com metade dos dados do RoBERTa.23 |
| **DeBERTa-V3-Large** | RTD (estilo ELECTRA) + Atenção Desagregada + GDES. 22 | SentencePiece (128k) | Não especificado 12 | Não especificado 12 | Não especificado 12 | L=24, H=1024, A=16, E=1024, P=~435M (304M backbone) 12 | Treinado com 160GB de dados (mesmo que V2).12 Custo computacional alto.106 |
| **alBERTina-1.5B-PTPT** | Pré-treinamento (continuado de DeBERTaV2-xxlarge) em Português PT-PT; Arquitetura DeBERTa. 122 | SentencePiece (128k) | 1e-5 (900M) 3 | 896 3 | 250k+80k+60k Passos 124 | L=48, H=1536, A=Não esp., E=Não esp., P=1.5B 124 | 96 horas / 16 GPUs A100 (40GB) 124 |
| **ModernBERT-Large** | MLM + RoPE + Atenção Alternada + GeGLU + Unpadding + Flash Attention 2; Treinado em 2T tokens (Inglês+Código). 25 | Vocab 50k | Agendamento Trapezoidal 130 | Não especificado | Treinado em 2T tokens 130 | L=28, H=1024, A=16 (provável), E=1024, P=395M 126 | Tempo não especificado / 8 GPUs H100 130 |
| **Humano** | Aquisição de linguagem natural ao longo da vida via exposição e interação. [Query point (9)] | Processamento multinível | Não aplicável | Não aplicável | Anos (~20+) | Cérebro humano | Anos / Cérebro humano |

**Nota:** L=Camadas, H=Dimensão Escondida, A=Cabeças de Atenção, E=Dimensão do Embedding (quando diferente de H), P=Parâmetros.

**4. Discussão: Evolução e Trade-offs**

A análise comparativa dos modelos revela uma trajetória evolutiva fascinante desde o BERT, marcada por diferentes focos de otimização e inovação, resultando em uma diversidade de modelos com trade-offs distintos.

Um dos principais eixos de desenvolvimento tem sido o balanço entre **eficiência e performance**. Enquanto o BERT original estabeleceu um alto padrão de desempenho, seu custo computacional motivou a criação de alternativas mais leves. DistilBERT 54 e ALBERT 18 exemplificam essa busca por eficiência. DistilBERT, através da destilação de conhecimento, reduziu o número de camadas e parâmetros, resultando em inferência significativamente mais rápida com uma perda de performance relativamente pequena.17 ALBERT, por sua vez, utilizou fatorização de embedding e compartilhamento de parâmetros para diminuir drasticamente a contagem de parâmetros e o consumo de memória, embora sem uma redução proporcional no tempo de computação por passo.74 Em contraste, modelos como RoBERTa 15, ELECTRA 21, DeBERTa 23, e ModernBERT 26 priorizaram alcançar ou superar o estado da arte em performance, muitas vezes à custa de maior complexidade arquitetônica ou requisitos computacionais de treinamento elevados, embora ELECTRA e DeBERTaV3 também tenham introduzido melhorias na eficiência de *amostra* durante o pré-treinamento. Técnicas como RTD 20 e otimizações como RoPE e Flash Attention 25 mostram que eficiência e performance não são necessariamente mutuamente exclusivas, mas representam um espectro de design onde diferentes modelos ocupam posições distintas.

Paralelamente, houve uma **evolução notável nos objetivos de pré-treinamento**. O BERT iniciou com a combinação de MLM e NSP.28 RoBERTa rapidamente questionou a utilidade do NSP, removendo-o e focando em otimizar o MLM com mascaramento dinâmico.15 ALBERT substituiu NSP por SOP, buscando uma tarefa inter-sentencial mais significativa.18 ELECTRA propôs uma mudança mais radical com o RTD, deslocando o foco da geração (predição de tokens mascarados) para a discriminação (identificação de tokens substituídos), o que se provou mais eficiente em termos de amostra.20 DistilBERT combinou MLM com perdas de destilação para transferir conhecimento.16 DeBERTaV3 adotou o RTD, combinando-o com sua arquitetura avançada.22 Essa trajetória demonstra uma exploração contínua de como extrair o máximo de informação de dados não rotulados de forma auto-supervisionada, indo além da simples previsão de palavras mascaradas.

As **inovações arquitetônicas** também foram cruciais. Enquanto RoBERTa e ELECTRA mantiveram a arquitetura Transformer do BERT relativamente intacta, DeBERTa introduziu a atenção desagregada 23, modificando fundamentalmente como as informações de conteúdo e posição interagem dentro do mecanismo de atenção. ModernBERT 25 representa a mais recente onda de modernização arquitetônica, importando técnicas como RoPE (para melhor tratamento de posições e sequências longas), atenção alternada (para eficiência em longo contexto) e ativações GeGLU, todas popularizadas por LLMs recentes. Essas mudanças visam não apenas melhorar a performance bruta, mas também abordar limitações como o comprimento de contexto restrito dos modelos anteriores e otimizar a eficiência computacional.

Não se pode ignorar o **papel crescente da escala de dados e do poder computacional**. RoBERTa demonstrou o impacto de treinar com 10x mais dados que o BERT.46 Modelos subsequentes continuaram essa tendência, com DeBERTaV3 usando 160GB 12 e ModernBERT sendo treinado em 2 trilhões de tokens.25 Da mesma forma, os recursos de hardware necessários para treinar esses modelos no estado da arte cresceram exponencialmente, desde os 64 chips TPU do BERT-Large 5 até as 1024 GPUs V100 do RoBERTa-Large 39 e as 8 H100s do ModernBERT.130 Isso levanta questões sobre a acessibilidade e a democratização da pesquisa em PLN 21, embora esforços como DistilBERT e ALBERT busquem oferecer alternativas mais eficientes.

Observa-se também uma **convergência de ideias** bem-sucedidas. DeBERTaV3 é um exemplo claro, unindo a arquitetura DeBERTa com o pré-treinamento RTD do ELECTRA, e necessitando da inovação GDES para fazer a combinação funcionar.22 ModernBERT aplica explicitamente técnicas desenvolvidas para LLMs decoders em um modelo encoder.26 alBERTina utiliza uma arquitetura SOTA em inglês (DeBERTa) como base para um modelo português.122 Isso sugere um amadurecimento do campo, onde os pesquisadores não seguem apenas uma linha evolutiva, mas combinam seletivamente os componentes mais promissores de diferentes abordagens (arquitetura, tarefa de pré-treinamento, otimizações de eficiência) para criar modelos ainda melhores.

Finalmente, a análise revela um espectro entre **especialização e generalização**. Modelos como BERT, RoBERTa, ELECTRA, DeBERTa e ModernBERT aspiram ser modelos fundacionais de propósito geral, pré-treinados em dados massivos e diversificados para servirem como base para uma ampla gama de tarefas downstream.4 Por outro lado, DistilBERT e ALBERT focam na eficiência, otimizando para cenários de implantação com recursos limitados.16 alBERTina representa a especialização linguística, adaptando uma arquitetura poderosa para uma língua específica (Português).122 Essa diversidade reflete as múltiplas necessidades do ecossistema de PLN, onde não existe um único "melhor" modelo, mas sim diferentes modelos otimizados para diferentes objetivos e restrições.

**5. Conclusão**

A evolução dos modelos de linguagem desde o BERT tem sido marcada por uma busca incessante por maior desempenho, eficiência e capacidade. A jornada começou com otimizações diretas do regime de treinamento do BERT, como demonstrado pelo RoBERTa, que destacou a importância da escala de dados, hiperparâmetros e a revisão dos objetivos de pré-treinamento originais. Simultaneamente, a necessidade de modelos mais leves e rápidos para aplicações práticas impulsionou o desenvolvimento de técnicas como a destilação de conhecimento (DistilBERT) e a redução paramétrica inteligente (ALBERT).

Paralelamente, a comunidade explorou objetivos de pré-treinamento alternativos ao MLM, com o RTD do ELECTRA emergindo como uma abordagem notavelmente mais eficiente em termos de amostra. Inovações na própria arquitetura Transformer, como a atenção desagregada do DeBERTa e as modernizações incorporadas no ModernBERT (RoPE, atenção alternada, GeGLU), demonstraram que ainda havia espaço para melhorias fundamentais no núcleo desses modelos. Modelos mais recentes, como DeBERTaV3 e ModernBERT, exemplificam a tendência de sintetizar as melhores ideias de diferentes linhagens de pesquisa, combinando arquiteturas avançadas com metodologias de treinamento eficientes e incorporando avanços de outras áreas, como LLMs. A especialização linguística, como vista no alBERTina, também se consolidou como uma direção importante para melhorar o desempenho em línguas específicas.

Os principais trade-offs que moldaram essa evolução incluem a tensão entre performance máxima e custo computacional/paramétrico, a busca por objetivos de pré-treinamento mais eficazes e eficientes, e o equilíbrio entre modelos de propósito geral e modelos especializados (seja por eficiência ou por língua/domínio). O estado da arte atual, representado por modelos como DeBERTaV3 e ModernBERT, reflete um alto grau de sofisticação tanto na arquitetura quanto no treinamento, alcançando performances notáveis em benchmarks estabelecidos e expandindo capacidades, como o tratamento de contextos mais longos. As direções futuras provavelmente continuarão a explorar esses trade-offs, buscando maior eficiência, melhor compreensão de contextos extensos, adaptação mais robusta a diversas línguas e domínios, e talvez novas arquiteturas ou paradigmas de pré-treinamento que transcendam as abordagens atuais.

#### Referências citadas

1. PALBERT - arXiv, acessado em abril 14, 2025, <https://arxiv.org/pdf/2204.03276>
2. ELECTRA for TensorFlow2 - NGC Catalog - NVIDIA, acessado em abril 14, 2025, <https://catalog.ngc.nvidia.com/orgs/nvidia/resources/electra_for_tensorflow2>
3. PORTULAN/albertina-900m-portuguese-ptpt-encoder - Hugging Face, acessado em abril 14, 2025, <https://huggingface.co/PORTULAN/albertina-900m-portuguese-ptpt-encoder>
4. [1810.04805] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding - arXiv, acessado em abril 14, 2025, <https://arxiv.org/abs/1810.04805>
5. arxiv.org, acessado em abril 14, 2025, <https://arxiv.org/pdf/1810.04805>
6. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, acessado em abril 14, 2025, <https://aclanthology.org/N19-1423/>
7. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding | Request PDF - ResearchGate, acessado em abril 14, 2025, <https://www.researchgate.net/publication/328230984_BERT_Pre-training_of_Deep_Bidirectional_Transformers_for_Language_Understanding>
8. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, acessado em abril 14, 2025, <https://research.google/pubs/bert-pre-training-of-deep-bidirectional-transformers-for-language-understanding/>
9. Evaluating Deep Learning Techniques for Natural Language Inference - MDPI, acessado em abril 14, 2025, <https://www.mdpi.com/2076-3417/13/4/2577>
10. Open Sourcing BERT: State-of-the-Art Pre-training for Natural Language Processin, acessado em abril 14, 2025, <https://research.google/blog/open-sourcing-bert-state-of-the-art-pre-training-for-natural-language-processing/>
11. The 17th Workshop on Building and Using Comparable Corpora @LREC-COLING-2024 - ACL Anthology, acessado em abril 14, 2025, <https://aclanthology.org/2024.bucc-1.pdf>
12. microsoft/deberta-v3-large · Hugging Face, acessado em abril 14, 2025, <https://huggingface.co/microsoft/deberta-v3-large>
13. [1907.11692] RoBERTa: A Robustly Optimized BERT Pretraining Approach - arXiv, acessado em abril 14, 2025, <https://arxiv.org/abs/1907.11692>
14. New Google Brain Optimizer Reduces BERT Pre-Training Time From Days to Minutes, acessado em abril 14, 2025, <https://syncedreview.com/2019/04/04/new-google-brain-optimizer-reduces-bert-pre-training-time-from-days-to-minutes/>
15. [1907.11692] RoBERTa: A Robustly Optimized BERT Pretraining Approach - ar5iv - arXiv, acessado em abril 14, 2025, <https://ar5iv.labs.arxiv.org/html/1907.11692>
16. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter - arXiv, acessado em abril 14, 2025, <https://arxiv.org/abs/1910.01108>
17. Distilbert: A Smaller, Faster, and Distilled BERT - Zilliz Learn, acessado em abril 14, 2025, <https://zilliz.com/learn/distilbert-distilled-version-of-bert>
18. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations - arXiv, acessado em abril 14, 2025, <https://arxiv.org/abs/1909.11942>
19. arXiv:1909.11942v6 [cs.CL] 9 Feb 2020, acessado em abril 14, 2025, <https://arxiv.org/pdf/1909.11942>
20. Maximizing Efficiency of Language Model Pre-training for Learning Representation - arXiv, acessado em abril 14, 2025, <https://arxiv.org/pdf/2110.06620>
21. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators, acessado em abril 14, 2025, <https://openreview.net/forum?id=r1xMH1BtvB>
22. DeBERTaV3: Improving DeBERTa using ELECTRA-Style Pre-Training with Gradient-Disentangled Embedding Sharing - arXiv, acessado em abril 14, 2025, <https://arxiv.org/abs/2111.09543>
23. deberta: decoding-enhanced bert with dis - arXiv, acessado em abril 14, 2025, <https://arxiv.org/pdf/2006.03654>
24. [2006.03654] DeBERTa: Decoding-enhanced BERT with Disentangled Attention - arXiv, acessado em abril 14, 2025, <https://arxiv.org/abs/2006.03654>
25. ModernBERT - Hugging Face, acessado em abril 14, 2025, <https://huggingface.co/docs/transformers/model_doc/modernbert>
26. Finally, a Replacement for BERT: Introducing ModernBERT - Hugging Face, acessado em abril 14, 2025, <https://huggingface.co/blog/modernbert>
27. Devlin, J., Chang, M.W., Lee, K., et al. (2018) Bert Pre-Training of Deep Bidirectional Transformers for Language Understanding. arXiv 1810.04805. - References - Scientific Research Publishing, acessado em abril 14, 2025, <https://www.scirp.org/reference/referencespapers?referenceid=3492350>
28. google-bert/bert-large-uncased · Hugging Face, acessado em abril 14, 2025, <https://huggingface.co/google-bert/bert-large-uncased>
29. google-bert/bert-base-uncased - Hugging Face, acessado em abril 14, 2025, <https://huggingface.co/google-bert/bert-base-uncased>
30. google-bert/bert-base-cased - Hugging Face, acessado em abril 14, 2025, <https://huggingface.co/google-bert/bert-base-cased>
31. BERT 101 - State Of The Art NLP Model Explained - Hugging Face, acessado em abril 14, 2025, <https://huggingface.co/blog/bert-101>
32. BERT - Hugging Face, acessado em abril 14, 2025, <https://huggingface.co/docs/transformers/model_doc/bert>
33. DistilBERT - Hugging Face, acessado em abril 14, 2025, <https://huggingface.co/docs/transformers/model_doc/distilbert>
34. Fine-tuning BERT with sequences longer than 512 tokens - Models - Hugging Face Forums, acessado em abril 14, 2025, <https://discuss.huggingface.co/t/fine-tuning-bert-with-sequences-longer-than-512-tokens/12652>
35. RoBERTa: A Robustly Optimized BERT Pretraining Approach - ResearchGate, acessado em abril 14, 2025, <https://www.researchgate.net/publication/334735779_RoBERTa_A_Robustly_Optimized_BERT_Pretraining_Approach>
36. RoBERTa - Hugging Face, acessado em abril 14, 2025, <https://huggingface.co/docs/transformers/model_doc/roberta>
37. fairseq/examples/roberta/README.md at main - GitHub, acessado em abril 14, 2025, <https://github.com/facebookresearch/fairseq/blob/main/examples/roberta/README.md>
38. Liu, Y., Ott, M., Goyal, N., et al. (2020) RoBERTa A Robustly Optimized BERT Pretraining Approach. - References - Scientific Research Publishing, acessado em abril 14, 2025, <https://www.scirp.org/reference/referencespapers?referenceid=3023380>
39. arxiv.org, acessado em abril 14, 2025, <https://arxiv.org/pdf/1907.11692>
40. RoBERTa- A Robustly Optimized BERT Pretraining Approach - I'm Hyunyoung2, acessado em abril 14, 2025, <https://hyunyoung2.github.io/2021/03/15/RoBERTa/>
41. RoBERTa: A Robustly Optimized BERT Pretraining Approach | BibSonomy, acessado em abril 14, 2025, <https://www.bibsonomy.org/bibtex/040474bcd625e7dcc649bb20c81104d2>
42. RoBERTa: An optimized method for pretraining self-supervised NLP systems - Meta AI, acessado em abril 14, 2025, <https://ai.meta.com/blog/roberta-an-optimized-method-for-pretraining-self-supervised-nlp-systems/>
43. RoBERTa: A Robustly Optimized BERT Pretraining Approach - AI Resources - Modular, acessado em abril 14, 2025, <https://www.modular.com/ai-resources/roberta-a-robustly-optimized-bert-pretraining-approach>
44. FacebookAI/roberta-large-mnli - Hugging Face, acessado em abril 14, 2025, <https://huggingface.co/FacebookAI/roberta-large-mnli>
45. FacebookAI/roberta-large - Hugging Face, acessado em abril 14, 2025, <https://huggingface.co/FacebookAI/roberta-large>
46. Pre-training BERT/RoBERTa language model using domain text, how long it gonna take estimately? which is faster? - Stack Overflow, acessado em abril 14, 2025, <https://stackoverflow.com/questions/60137162/pre-training-bert-roberta-language-model-using-domain-text-how-long-it-gonna-ta>
47. [R][1907.11692] RoBERTa: A Robustly Optimized BERT Pretraining Approach - Reddit, acessado em abril 14, 2025, <https://www.reddit.com/r/MachineLearning/comments/cjbcxm/r190711692_roberta_a_robustly_optimized_bert/>
48. FacebookAI/roberta-base - Hugging Face, acessado em abril 14, 2025, <https://huggingface.co/FacebookAI/roberta-base>
49. Effects of Pre-training and Fine-tuning Time on the Linear Connectivity of Language Models for Natural Language Inference - Stanford University, acessado em abril 14, 2025, <https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1244/final-projects/KushalThaman.pdf>
50. RoBERTa — transformers 3.0.2 documentation - Hugging Face, acessado em abril 14, 2025, <https://huggingface.co/transformers/v3.0.2/model_doc/roberta.html>
51. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter, acessado em abril 14, 2025, <https://ysu1989.github.io/courses/au20/cse5539/DistilBERT.pdf>
52. RoBERTa: A Robustly Optimized BERT Pretraining Approach | OpenReview, acessado em abril 14, 2025, <https://openreview.net/forum?id=SyxS0T4tvS>
53. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations, acessado em abril 14, 2025, <https://openreview.net/forum?id=H1eA7AEtvS>
54. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter - ResearchGate, acessado em abril 14, 2025, <https://www.researchgate.net/publication/336230729_DistilBERT_a_distilled_version_of_BERT_smaller_faster_cheaper_and_lighter>
55. DistilBERT - Hugging Face, acessado em abril 14, 2025, <https://huggingface.co/docs/transformers/main/en//model_doc/distilbert>
56. DistilBERT - Hugging Face, acessado em abril 14, 2025, <https://huggingface.co/docs/transformers/v4.34.0/model_doc/distilbert>
57. DistilBERT Explained - Papers With Code, acessado em abril 14, 2025, <https://paperswithcode.com/method/distillbert>
58. DistilBERT — distilled version of BERT - KiKaBeN, acessado em abril 14, 2025, <https://kikaben.com/distilbert-distilled-version-of-bert/>
59. Contrastive Learning in Distilled Models - arXiv, acessado em abril 14, 2025, <https://arxiv.org/pdf/2401.12472>
60. HuggingFace DistilBERT - Scaler Topics, acessado em abril 14, 2025, <https://www.scaler.com/topics/nlp/distilbert/>
61. Knowledge Distillation with Vision Transformers - Hugging Face Community Computer Vision Course, acessado em abril 14, 2025, <https://huggingface.co/learn/computer-vision-course/unit3/vision-transformers/knowledge-distillation>
62. Hugging Face Distillation Methods Tutorial | Restackio, acessado em abril 14, 2025, <https://www.restack.io/p/model-distillation-answer-hugging-face-tutorial-cat-ai>
63. How to Use DistilBERT: A Comprehensive Guide - BytePlus, acessado em abril 14, 2025, <https://www.byteplus.com/en/topic/400600>
64. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter - arXiv, acessado em abril 14, 2025, <https://arxiv.org/pdf/1910.01108>
65. BERTino: an Italian DistilBERT model - CEUR-WS.org, acessado em abril 14, 2025, <https://ceur-ws.org/Vol-2769/paper_09.pdf>
66. This work focuses on the efficiency of knowledge distillation in generating a lightweight but strong BERT-based model for natural language processing (NLP) applications and specific attention on detecting Attention deficit hyperactivity disorder (ADHD) - arXiv, acessado em abril 14, 2025, <https://arxiv.org/html/2411.00052v1>
67. Towards Transfer Learning Techniques—BERT, DistilBERT, BERTimbau, and DistilBERTimbau for Automatic Text Classification from Different Languages: A Case Study - MDPI, acessado em abril 14, 2025, <https://www.mdpi.com/1424-8220/22/21/8184>
68. acessado em dezembro 31, 1969, <https://arxiv.org/pdf/1910.01108.pdf>
69. distilbert/distilbert-base-uncased · Hugging Face, acessado em abril 14, 2025, <https://huggingface.co/distilbert/distilbert-base-uncased>
70. mALBERT: Is a Compact Multilingual BERT Model Still Worth It? - arXiv, acessado em abril 14, 2025, <https://arxiv.org/html/2403.18338v1>
71. Exploring Internal Numeracy in Language Models: A Case Study on ALBERT - arXiv, acessado em abril 14, 2025, <https://arxiv.org/html/2404.16574>
72. arXiv:2204.09145v2 [cs.CL] 25 Jan 2023, acessado em abril 14, 2025, <https://arxiv.org/pdf/2204.09145>
73. arXiv:2404.16574v1 [cs.CL] 25 Apr 2024, acessado em abril 14, 2025, <https://arxiv.org/pdf/2404.16574>
74. Review - ALBERT: A Lite BERT for Self-supervised Learning of Language Representations, acessado em abril 14, 2025, <https://www.assemblyai.com/blog/review-albert>
75. BERT vs ALBERT explained - Papers With Code, acessado em abril 14, 2025, <https://paperswithcode.com/paper/bert-vs-albert-explained>
76. ALBERT - Hugging Face, acessado em abril 14, 2025, <https://huggingface.co/docs/transformers/model_doc/albert>
77. ALBERT — transformers 2.9.1 documentation - Hugging Face, acessado em abril 14, 2025, <https://huggingface.co/transformers/v2.9.1/model_doc/albert.html>
78. ALBERT — transformers 3.0.2 documentation - Hugging Face, acessado em abril 14, 2025, <https://huggingface.co/transformers/v3.0.2/model_doc/albert.html>
79. albert/albert-xxlarge-v2 · Hugging Face, acessado em abril 14, 2025, <https://huggingface.co/albert/albert-xxlarge-v2>
80. README.md · albert/albert-xxlarge-v2 at main - Hugging Face, acessado em abril 14, 2025, <https://huggingface.co/albert/albert-xxlarge-v2/blame/main/README.md>
81. Create ALBERT XXLarge v2 model card - Hugging Face, acessado em abril 14, 2025, <https://huggingface.co/albert/albert-xxlarge-v2/commit/aaec31cf649a4d91a96b11f83eb5b2985eaf8ee5>
82. tftransformers/albert-xxlarge-v2 - Hugging Face, acessado em abril 14, 2025, <https://huggingface.co/tftransformers/albert-xxlarge-v2>
83. acessado em dezembro 31, 1969, <https://arxiv.org/pdf/1909.11942.pdf>
84. [ALBERT] How to deal with Model diverged with loss = NaN when training from scratch? #111 - GitHub, acessado em abril 14, 2025, <https://github.com/google-research/albert/issues/111>
85. google-research/albert: ALBERT: A Lite BERT for Self-supervised Learning of Language Representations - GitHub, acessado em abril 14, 2025, <https://github.com/google-research/albert>
86. Train ALBERT for natural language processing with TensorFlow on Amazon SageMaker, acessado em abril 14, 2025, <https://aws.amazon.com/blogs/machine-learning/train-albert-for-natural-language-processing-with-tensorflow-on-amazon-sagemaker/>
87. Pretrained Language Model Embryology: The Birth of ALBERT - ACL Anthology, acessado em abril 14, 2025, <https://aclanthology.org/2020.emnlp-main.553.pdf>
88. ALBERT on Determined: Distributed Training with Spot Instances, acessado em abril 14, 2025, <https://www.determined.ai/blog/albert-spot-benchmark>
89. [2106.00139] Training ELECTRA Augmented with Multi-word Selection - ar5iv - arXiv, acessado em abril 14, 2025, <https://ar5iv.labs.arxiv.org/html/2106.00139>
90. [2207.08141] ELECTRA is a Zero-Shot Learner, Too - ar5iv, acessado em abril 14, 2025, <https://ar5iv.labs.arxiv.org/html/2207.08141>
91. arXiv:2106.07176v4 [cs.CL] 7 Jun 2022, acessado em abril 14, 2025, <https://par.nsf.gov/servlets/purl/10351302>
92. Are ELECTRA's Sentence Embeddings Beyond Repair? The Case of Semantic Textual Similarity - arXiv, acessado em abril 14, 2025, <https://arxiv.org/html/2402.13130v1>
93. arXiv:2305.12567v1 [cs.CL] 21 May 2023, acessado em abril 14, 2025, <https://arxiv.org/pdf/2305.12567>
94. google/electra-large-generator - Hugging Face, acessado em abril 14, 2025, <https://huggingface.co/google/electra-large-generator>
95. Exploring ELECTRA - Efficient Pre-training for Transformers - DEV Community, acessado em abril 14, 2025, <https://dev.to/nareshnishad/exploring-electra-efficient-pre-training-for-transformers-o7j>
96. google/electra-large-discriminator · Hugging Face, acessado em abril 14, 2025, <https://huggingface.co/google/electra-large-discriminator>
97. google-research/electra: ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators - GitHub, acessado em abril 14, 2025, <https://github.com/google-research/electra>
98. PEER: Pre-training ELECTRA Extended by Ranking - ACL Anthology, acessado em abril 14, 2025, <https://aclanthology.org/2023.findings-acl.405.pdf>
99. Efficient transfer learning for NLP with ELECTRA - arXiv, acessado em abril 14, 2025, <https://arxiv.org/pdf/2104.02756>
100. ELECTRA: PRE-TRAINING TEXT ENCODERS AS DIS- CRIMINATORS RATHER THAN GENERATORS - OpenReview, acessado em abril 14, 2025, <https://openreview.net/attachment?id=r1xMH1BtvB&name=original_pdf>
101. ELECTRA - Hugging Face, acessado em abril 14, 2025, <https://huggingface.co/docs/transformers/v4.13.0/model_doc/electra>
102. ELECTRA — transformers 3.0.2 documentation - Hugging Face, acessado em abril 14, 2025, <https://huggingface.co/transformers/v3.0.2/model_doc/electra.html>
103. ELECTRA - Hugging Face, acessado em abril 14, 2025, <https://huggingface.co/docs/transformers/model_doc/electra>
104. (PDF) DeBERTaV3: Improving DeBERTa using ELECTRA-Style Pre-Training with Gradient-Disentangled Embedding Sharing - ResearchGate, acessado em abril 14, 2025, <https://www.researchgate.net/publication/356375694_DeBERTaV3_Improving_DeBERTa_using_ELECTRA-Style_Pre-Training_with_Gradient-Disentangled_Embedding_Sharing>
105. Understand and Modularize Generator Optimization in ELECTRA-style Pretraining - OpenReview, acessado em abril 14, 2025, <https://openreview.net/pdf?id=ikE60aXe8M>
106. DeBERTaV3: Improving DeBERTa using ELECTRA-Style Pre-Training with Gradient-Disentangled Embedding Sharing | OpenReview, acessado em abril 14, 2025, <https://openreview.net/forum?id=sE7-XhLxHA>
107. arXiv:2301.10439v2 [cs.CL] 10 Feb 2023, acessado em abril 14, 2025, <https://arxiv.org/pdf/2301.10439>
108. Instruct-DeBERTa: A Hybrid Approach for Aspect-based Sentiment Analysis on Textual Reviews - arXiv, acessado em abril 14, 2025, <http://arxiv.org/pdf/2408.13202>
109. [2006.03654] DeBERTa: Decoding-enhanced BERT with Disentangled Attention - ar5iv, acessado em abril 14, 2025, <https://ar5iv.labs.arxiv.org/html/2006.03654>
110. The implementation of DeBERTa - GitHub, acessado em abril 14, 2025, <https://github.com/microsoft/DeBERTa>
111. Albertina PT-PT - PORTULAN CLARIN, acessado em abril 14, 2025, <https://portulanclarin.net/repository/browse/albertina-pt-pt/580747f8005c11eeb87402420a870130b0dff31356c24c9f91e5df5c9990074a/>
112. Albertina PT-\* models - PORTULAN CLARIN, acessado em abril 14, 2025, <https://portulanclarin.net/models/>
113. advancing neural encoding of portuguese - arXiv, acessado em abril 14, 2025, <https://arxiv.org/pdf/2305.06721>
114. microsoft/mdeberta-v3-base - Hugging Face, acessado em abril 14, 2025, <https://huggingface.co/microsoft/mdeberta-v3-base>
115. Improve README · microsoft/deberta-v3-large at ba88b9f - Hugging Face, acessado em abril 14, 2025, <https://huggingface.co/microsoft/deberta-v3-large/commit/ba88b9f7a65194e583ec1a516279f9641b864267>
116. Deberta V3 Large · Models - Dataloop, acessado em abril 14, 2025, <https://dataloop.ai/library/model/microsoft_deberta-v3-large/>
117. [2111.09543] DeBERTaV3: Improving DeBERTa using ELECTRA-Style Pre-Training with Gradient-Disentangled Embedding Sharing - ar5iv, acessado em abril 14, 2025, <https://ar5iv.labs.arxiv.org/html/2111.09543>
118. mrm8488/deberta-v3-large-finetuned-mnli - Hugging Face, acessado em abril 14, 2025, <https://huggingface.co/mrm8488/deberta-v3-large-finetuned-mnli>
119. Pretraining the deberta-v3 by larger context length. · Issue #153 - GitHub, acessado em abril 14, 2025, <https://github.com/microsoft/DeBERTa/issues/153>
120. MosaicBERT: A Bidirectional Encoder Optimized for Fast Pretraining - NIPS papers, acessado em abril 14, 2025, <https://papers.nips.cc/paper_files/paper/2023/file/095a6917768712b7ccc61acbeecad1d8-Paper-Conference.pdf>
121. DeBERTa - Hugging Face, acessado em abril 14, 2025, <https://huggingface.co/docs/transformers/main/model_doc/deberta>
122. Advancing Neural Encoding of - Portuguese with Transformer Albertina PT, acessado em abril 14, 2025, <https://www.di.fc.ul.pt/~ahb/pubs/2023RodriguesGomesEtAl.pdf>
123. Advancing Neural Encoding of Portuguese with Transformer Albertina PT-\* - arXiv, acessado em abril 14, 2025, <https://arxiv.org/html/2305.06721>
124. Update README.md · PORTULAN/albertina-1b5-portuguese-ptpt-encoder at c8274c0, acessado em abril 14, 2025, <https://huggingface.co/PORTULAN/albertina-1b5-portuguese-ptpt-encoder/commit/c8274c01a6bedaa5117c7aadce001d32e857e580>
125. Understanding ModernBERT: The Future of Efficient Language Processing, acessado em abril 14, 2025, <https://apxml.com/posts/modernbert-nlp-encoder>
126. ModernBERT: A New Star in Natural Language Processing - Tech Explorer, acessado em abril 14, 2025, <https://stable-learn.com/en/modernbert-analysis/>
127. Unlocking RAG's Potential with ModernBERT - Analytics Vidhya, acessado em abril 14, 2025, <https://www.analyticsvidhya.com/blog/2025/01/unlocking-rags-potential-with-modernbert/>
128. ModernBERT: Redefining NLP with Advanced Transformer Models - Metadesign Solutions, acessado em abril 14, 2025, <https://metadesignsolutions.com/modern-bert-redefining-nlp-with-advanced-transformer-models/>
129. Mastering ModernBERT: The Evolution of Encoder Models - Association of Data Scientists, acessado em abril 14, 2025, <https://adasci.org/mastering-modernbert-the-evolution-of-encoder-models/>
130. answerdotai/ModernBERT-base - Hugging Face, acessado em abril 14, 2025, <https://huggingface.co/answerdotai/ModernBERT-base>
131. arXiv:2412.13663v2 [cs.CL] 19 Dec 2024, acessado em abril 14, 2025, <https://arxiv.org/pdf/2412.13663>
132. Enhancing Sentiment Analysis with ModernBERT - Analytics Vidhya, acessado em abril 14, 2025, <https://www.analyticsvidhya.com/blog/2025/01/sentiment-analysis-using-modernbert/>
133. Smarter, Better, Faster, Longer: A Modern Bidirectional Encoder for Fast, Memory Efficient, and Long Context Finetuning and Inference - arXiv, acessado em abril 14, 2025, <https://arxiv.org/html/2412.13663v2>
134. ModernBERT vs LLMs for Detecting Adverse Drug Reactions - Paul Simmering, acessado em abril 14, 2025, <https://simmering.dev/blog/modernbert-vs-llm/>
135. Should I try multiple optimizers when fine-tuning pre-trained Transformers for NLP tasks? Should I tune their hyperparameters? - ACL Anthology, acessado em abril 14, 2025, <https://aclanthology.org/2024.eacl-long.157.pdf>
136. Surge Phenomenon in Optimal Learning Rate and Batch Size Scaling - arXiv, acessado em abril 14, 2025, <https://arxiv.org/html/2405.14578v5>