Otimização de Hiperparâmetros de Modelos Transformer via Meta-heurísticas de Otimização: Um Estudo com Detecção de Sinais de Depressão em Tweets

Bianca Lopes dos Santos
Engenharia de Sistemas
Universidade Federal de Minas Gerais
Belo Horizonte, Brasil
biancalds18@gmail.com

Lucas Pimenta Braga
Engenharia de Sistemas
Universidade Federal de Minas Gerais
Belo Horizonte, Brasil
lucaspimentabraga@gmail.com

***Abstract*—A depressão é um transtorno de saúde mental de alta prevalência, e sinais precoces frequentemente aparecem em interações espontâneas em redes sociais. Isso tem motivado o uso de modelos de Processamento de Linguagem Natural (PLN) para apoiar a detecção automática de indícios de depressão em textos curtos, como *tweets* [1], [2]. Entre essas abordagens, modelos Transformer consolidaram-se como a arquitetura de referência, mas seu desempenho é extremamente sensível à escolha de hiperparâmetros, cuja calibração manual é custosa, pouco reprodutível e frequentemente subótima [3]–[5].**

Neste trabalho, investiga-se o uso de meta-heurísticas estocásticas de otimização — incluindo um algoritmo evolutivo (Differential Evolution, DE), um método de inteligência de enxame (Particle Swarm Optimization, PSO) e um esquema de recozimento simulado (Simulated Annealing, SA) — para otimizar automaticamente os hiperparâmetros de um *encoder* Transformer leve aplicado à detecção binária de sinais de depressão em *tweets*. Considera-se um espaço de busca misto (dimensões do modelo, regularização e parâmetros de treinamento) e compara-se um modelo *baseline*, configurado manualmente, com as configurações produzidas por cada meta-heurística.

Os resultados mostram que todas as meta-heurísticas superam o baseline em acurácia e F1-score. O melhor modelo, obtido via DE, apresenta aumentos relativos de aproximadamente $11{,}7\,\%$ na acurácia, $18{,}9\,\%$ no F1-score e $40{,}1\,\%$ no *recall* da classe positiva no conjunto de teste, em relação ao baseline. Esses ganhos indicam que a otimização meta-heurística de hiperparâmetros constitui uma estratégia eficaz para refinar o desempenho de modelos Transformer compactos, evidenciando o papel de algoritmos evolutivos e afins como ferramentas práticas de ajuste em arquiteturas de atenção.

***Palavras-chave*—Meta-heurísticas, Differential Evolution, Particle Swarm Optimization, Simulated Annealing, Otimização de Hiperparâmetros, Transformers, PLN, Depressão.**

I. Introdução

Transformers [6], [7] se consolidaram como arquitetura padrão em Processamento de Linguagem Natural (PLN), alcançando desempenho de estado da arte em tarefas de classificação, tradução e extração de informação. Apesar disso, o desempenho final desses modelos depende criticamente de um conjunto extenso de hiperparâmetros: dimensões internas, número de camadas e cabeças de atenção, taxas de *dropout*, *weight decay*, tamanho do lote, taxa de aprendizado e comprimento máximo da sequência, entre outros. A calibração manual desses hiperparâmetros é cara, pouco reprodutível e tende a produzir configurações subótimas.

Estratégias clássicas de ajuste, como *grid search*, escalam de forma explosiva com o número de hiperparâmetros. Mesmo abordagens mais eficientes, como *random search* [3] e otimização bayesiana [4], costumam exigir dezenas a centenas de avaliações para explorar adequadamente o espaço de busca, o que nem sempre é viável em ambientes com recursos computacionais limitados [5], [8]. Esse cenário se agrava em Transformers, cujo custo de uma única avaliação (treinamento por poucas épocas) já é significativo.

Meta-heurísticas de otimização constituem uma alternativa natural nesse contexto. Algoritmos evolutivos, de enxame e métodos inspirados em processos físicos são capazes de realizar busca global em espaços de alta dimensão, tratando a função objetivo como uma *caixa-preta* e combinando variáveis contínuas, discretas e em escala logarítmica [9]. Em redes neurais profundas, esses métodos vêm sendo usados para escolher arquiteturas, ajustar hiperparâmetros e, em alguns casos, otimizar diretamente pesos [9], [10]. Em PLN, trabalhos recentes exploram meta-heurísticas para ajustar hiperparâmetros de modelos BERT e variantes [11], [12], indicando que abordagens estocásticas podem competir com técnicas mais tradicionais de otimização.

Neste artigo, investiga-se de forma sistemática o uso de meta-heurísticas para otimização de hiperparâmetros de um Transformer leve. O estudo possui ênfase na comparação entre algoritmos de diferentes famílias: um algoritmo evolutivo clássico (Differential Evolution, DE [13]), um método de inteligência de enxame (Particle Swarm Optimization, PSO [14]) e um algoritmo de trajetória inspirado em recozimento físico (Simulated Annealing, SA [15]). Como estudo de caso, utiliza-se um conjunto de *tweets* rotulados quanto à presença de sinais de depressão [1], [2], [16], [17], permitindo avaliar

o impacto das escolhas de hiperparâmetros em uma tarefa real de classificação binária sensível.

As principais contribuições deste trabalho são:

- desenvolvimento de um *pipeline* reprodutível para otimização de hiperparâmetros de um Transformer leve com meta-heurísticas;
- definição de um espaço de hiperparâmetros misto que combina parâmetros arquiteturais, de regularização e de treinamento, refletindo cenários reais de implementação de Transformers;
- comparação empírica entre DE, PSO e SA, quantificando ganhos de desempenho em relação a um *baseline* configurado manualmente;
- análise da melhor configuração encontrada, destacando o papel de cada grupo de hiperparâmetros para o desempenho.

II. Trabalhos Relacionados

Os trabalhos relacionados podem ser organizados em três eixos principais:

(i) detecção de depressão em mídias sociais com *deep learning*;
(ii) otimização de hiperparâmetros em modelos de aprendizado profundo;
(iii) uso de meta-heurísticas em PLN e modelos baseados em Transformers.

No primeiro eixo, Orabi et al. [1] utilizaram redes convolucionais e recorrentes para classificar usuários do Twitter quanto à presença de depressão, a partir de históricos de postagens. Trabalhos mais recentes exploram arquiteturas mais profundas e bases ampliadas, combinando diferentes características textuais [16]. Com a popularização de Transformers, surgiram abordagens baseadas em BERT e variantes para estimar presença e severidade de sintomas depressivos em textos de redes sociais [2], [17]. Neste artigo, esse eixo aparece como contexto e fonte de dados para a avaliação das meta-heurísticas.

No segundo eixo, Bergstra e Bengio [3] mostraram que *random search* pode ser mais eficiente que *grid search* em espaços de alta dimensão. Snoek et al. [4] introduziram a otimização bayesiana com processos gaussianos para hiperparâmetros de modelos de aprendizado de máquina, enquanto Li et al. [5] propuseram o Hyperband, baseado em *multi-armed bandits* e *early-stopping*. Revisões recentes sumarizam o estado da arte em técnicas de otimização de hiperparâmetros para redes profundas [8].

No terceiro eixo, meta-heurísticas têm sido aplicadas com sucesso ao ajuste de modelos de aprendizado profundo [9]. Han et al. [10] empregam Differential Evolution para melhorar o desempenho de modelos aplicados a dados de produção animal; Lorenzo et al. [18] utilizam PSO para seleção de hiperparâmetros em redes profundas; Zeng et al. [12] aplicam PSO a modelos de classificação de documentos e El-Demerdash et al. [11] investigam meta-heurísticas para otimizar hiperparâmetros de BERT em análise de sentimentos.

Entretanto, ainda há poucos estudos que combinam de forma explícita o uso de meta-heurísticas para otimização de hiperparâmetros em modelos Transformer com todos os seguintes elementos:

(i) um Transformer leve implementado diretamente, sem recorrer apenas a *checkpoints* pré-treinados;
(ii) uma comparação sistemática entre meta-heurísticas de famílias distintas (evolutivas, de enxame e de trajetória) aplicadas ao ajuste de hiperparâmetros.

Busca-se, neste trabalho, contribuir nessa lacuna por meio da avaliação conjunta de DE, PSO e SA na calibração de um Transformer leve sob forte restrição de orçamento de avaliações.

III. Metodologia

A metodologia integra, em um único *pipeline*, o pré-processamento textual, a modelagem com um encoder Transformer leve e a otimização de hiperparâmetros via meta-heurísticas estocásticas (Figura 1).

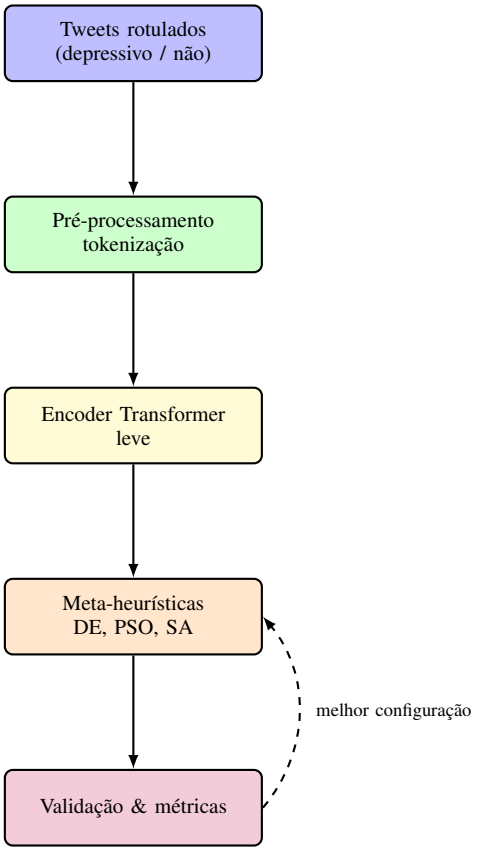

Figura 1. Fluxo geral do *pipeline* de otimização de hiperparâmetros.

A. Dados e pré-processamento

Utiliza-se, neste trabalho, o conjunto *Depression: Twitter Dataset + Feature Extraction*, disponibilizado por InFamousCoder no Kaggle [19]. A base é composta por aproximadamente 20 000 *tweets* em inglês, rotulados em duas classes

("depressed" e "non-depressed"), com distribuição aproximadamente balanceada. O conjunto foi originalmente construído a partir da API do Twitter e disponibilizado com atributos derivados relacionados a tópicos, emojis e outras características; neste estudo, emprega-se apenas o campo de texto dos *tweets* e o rótulo binário de classe, aplicando-se um pré-processamento próprio.

Sobre o texto, aplica-se um pré-processamento leve:

- conversão para minúsculas;
- tokenização baseada em espaços e pontuação;
- construção de vocabulário com frequência mínima de 2 ocorrências;
- inclusão de tokens especiais reservados para preenchimento e termos desconhecidos;
- truncamento ou preenchimento das sequências até um comprimento máximo definido.

Exemplo

Exemplo ilustrativo de sequência:
Frase original: `"não aguento mais essa rotina"`
Tokenização: `["não","aguento","mais","essa","rotina"]`
Índices no vocabulário (exemplo): `[15, 203, 87, 45, 310]`
Tokens especiais no vocabulário são reservados para preenchimento de sequências e termos não observados no treino.

A divisão em treino, validação e teste é feita por *split* estratificado 70/15/15, preservando a proporção de classes em cada subconjunto.

B. Modelagem: Transformer leve

A arquitetura do modelo é um Transformer encoder leve seguido de uma cabeça de classificação densa.

Embeddings: Cada índice da sequência é associado a um vetor contínuo de dimensão fixa. Assim, a frase tokenizada é convertida em uma matriz em que cada linha corresponde a um token em um espaço vetorial de maior dimensão.

Codificador Transformer: Sobre a sequência de embeddings são aplicadas camadas com atenção multi-cabeças combinadas a camadas *feed-forward* posicionais, produzindo uma representação contextualizada para cada token. Por exemplo, na frase "`eu NÃO estou feliz hoje`", o mecanismo de atenção tende a aprender que o termo "não" altera diretamente o sentido de "feliz", o que se reflete nos vetores associados a essas palavras.

Cabeça de classificação: Ao final do encoder, obtém-se uma sequência de vetores contextualizados. Essa sequência é agregada em um único vetor que resume o *tweet* e passa por camadas densas com função de ativação não linear, normalização e *dropout*. A última camada é uma unidade sigmoidal que produz uma probabilidade entre 0 e 1 para a classe positiva (tweet com sinais de depressão).

Exemplo

Exemplo ilustrativo da cabeça de classificação:
Vetor de entrada (saída agregada do encoder):
`[0.12, -0.80, 1.44, ..., 0.03]`
Após a camada densa (pesos + *bias* e função de ativação), obtém-se um escalar que, ao passar pela sigmoide, gera:
`Saída sigmoidal: 0.87`
Interpretação: o modelo atribui aproximadamente 87% de probabilidade de que o *tweet* pertença à classe com sinais de depressão.

C. Modelo baseline e métricas

Como ponto de partida para a otimização, define-se um modelo *baseline* configurado manualmente. A ideia é adotar uma arquitetura suficientemente expressiva para capturar padrões relevantes nos textos. Para isso, escolhe-se um encoder Transformer simples, com dimensões internas moderadas e regularização simples, treinado por poucas épocas.

A Tabela I resume os hiperparâmetros adotados nesse modelo de referência.

Tabela I
CONFIGURAÇÃO DO MODELO BASELINE

Hiperparâmetro	Valor
Dimensão dos embeddings	96
Número de cabeças de atenção	2
Número de camadas do encoder	1
Dimensão da camada *feed-forward*	256
Taxa de *dropout*	0,20
Tamanho do lote	64
Comprimento máx. da sequência	96
Taxa de aprendizado	3×10^{-4}
Weight decay	0
Número de épocas	3

Essa configuração busca equilibrar custo e capacidade: a dimensão dos embeddings e da camada intermediária foi escolhida para manter o número de parâmetros em nível moderado; uma única camada de atenção reduz a profundidade e o tempo de treinamento; e a taxa de *dropout* de 0,20 atua como regularizador simples, adequado para dados textuais de tamanho curto. A taxa de aprendizado e o tamanho do lote seguem valores típicos reportados na literatura para modelos Transformer de pequeno porte, ajustados empiricamente para garantir treinamento estável em 3 épocas.

No conjunto de teste, o baseline apresenta o desempenho mostrado na Tabela II.

Tabela II
DESEMPENHO DO MODELO BASELINE NO CONJUNTO DE TESTE

Métrica	Valor
Acurácia	0,7700
Precisão	0,8630
Recall	0,6364
F1-score	0,7326
Especificidade	0,9010

Observa-se que o modelo baseline apresenta boa precisão e alta especificidade, mas um *recall* mais baixo para a classe

positiva, resultando em F1-score intermediário. Esse comportamento indica que, embora o modelo tenda a errar pouco quando prevê a presença de sinais de depressão, ainda deixa de identificar uma fração relevante dos casos positivos, o que motiva a aplicação de técnicas de otimização de hiperparâmetros.

D. Espaço de hiperparâmetros

O espaço de hiperparâmetros explorado pelas meta-heurísticas foi definido para contemplar três grupos principais:

(i) parâmetros arquiteturais do encoder Transformer;
(ii) parâmetros de regularização;
(iii) parâmetros de treinamento.

A Tabela III apresenta os intervalos e tipos adotados.

Tabela III
ESPAÇO DE HIPERPARÂMETROS EXPLORADO

Parâmetro	**Tipo**	**Intervalo / opções**
Dim. embeddings	Discreto	$\{64, 96, 128, 160, 192, 224, 256\}$
Dim. *feed-forward*	Discreto	$\{256, 384, 512, 768, 1024\}$
Nº camadas do encoder	Discreto	$\{1, 2, 3, 4\}$
Nº cabeças de atenção	Discreto	$\{2, 4, 8\}$
Taxa de *dropout*	Contínuo	$[0{,}05, 0{,}50]$
Taxa de aprendizado	Log-contínuo	$[10^{-4}, 5 \cdot 10^{-3}]$
Weight decay	Log-contínuo	$[10^{-6}, 5 \cdot 10^{-4}]$
Tamanho do lote	Discreto	$\{32, 48, 64, 80, 96, 112, 128\}$
Comprimento máximo	Discreto	$\{64, 96, 128, 160, 192\}$

As faixas para dimensões internas (embeddings e camada *feed-forward*), número de camadas e número de cabeças de atenção permitem variar a capacidade representacional do modelo desde configurações bastante compactas até arquiteturas moderadamente maiores. Valores muito acima desses limites tenderiam a aumentar excessivamente o número de parâmetros e o tempo de treino por avaliação.

Os hiperparâmetros de regularização e treinamento são tratados em escala contínua ou logarítmica para favorecer a busca em regiões frequentemente críticas do espaço: taxas de *dropout* muito baixas podem levar a sobreajuste, enquanto taxas muito altas degradam a capacidade do modelo; a taxa de aprendizado influencia diretamente estabilidade e velocidade de convergência; o termo de *weight decay* atua como regularizador adicional sobre os pesos. O tamanho do lote afeta tanto a variância do gradiente quanto o uso de memória.

Ao combinar parâmetros discretos, contínuos e log-contínuos em um mesmo espaço de busca, cria-se um cenário desafiador para métodos determinísticos tradicionais, mas particularmente adequado para o uso de meta-heurísticas, que tratam a função objetivo como uma *caixa-preta* e conseguem explorar regiões promissoras do espaço de hiperparâmetros com um número limitado de avaliações.

IV. ALGORITMOS EVOLUCIONÁRIOS E META-HEURÍSTICAS

Para a otimização dos hiperparâmetros do modelo Transformer leve, foram empregadas três meta-heurísticas clássicas: Differential Evolution (DE), Particle Swarm Optimization (PSO) e Simulated Annealing (SA). A escolha desses métodos se fundamenta na sua capacidade de explorar espaços de busca de alta dimensionalidade, não lineares e potencialmente multimodais, como é o caso da configuração de modelos baseados em Transformers.

Cada algoritmo foi executado sob a mesma restrição orçamentária: um total de **12 avaliações** da função objetivo, em que cada avaliação corresponde ao treinamento do modelo por **3 épocas**. Essa limitação simula um cenário realista de otimização com recursos computacionais restritos, reforçando a importância do uso de meta-heurísticas eficientes e capazes de extrair o máximo de informação em poucas iterações.

A. Differential Evolution (DE)

O DE [13] é um algoritmo evolutivo populacional baseado em operações de mutação diferencial e recombinação entre vetores candidatos. A estratégia central consiste em gerar novos indivíduos combinando diferenças entre vetores da população, estimulando a exploração global do espaço de busca. Sua eficácia em otimização contínua o torna especialmente adequado para hiperparâmetros como taxa de aprendizado, *dropout* e *weight decay*, que apresentam comportamento sensível e não linear. Além disso, o DE possui poucos hiperparâmetros internos e, quando bem calibrado, converge rapidamente mesmo com populações pequenas.

B. Particle Swarm Optimization (PSO)

O PSO [14] opera por meio de um enxame de partículas que se deslocam no espaço de busca ajustando suas velocidades segundo dois componentes fundamentais: a melhor posição já encontrada por cada partícula e a melhor posição global identificada pelo enxame. Essa dinâmica de cooperação e competição permite ao PSO equilibrar exploração e intensificação. Em problemas de otimização de modelos de aprendizado profundo, o PSO se mostra eficiente para capturar interações complexas entre hiperparâmetros, sendo menos suscetível a ficar preso em mínimos locais quando comparado a métodos puramente determinísticos [12], [18].

C. Simulated Annealing (SA)

O SA [15] é um método estocástico inspirado no processo físico de recozimento térmico, no qual materiais aquecidos são gradualmente resfriados até atingirem um estado de baixa energia. No contexto de otimização, o algoritmo emprega um esquema de temperatura que decai ao longo do tempo, permitindo aceitar soluções piores com uma probabilidade controlada. Essa característica é essencial para escapar de mínimos locais nas fases iniciais da busca. À medida que a temperatura diminui, o processo se torna mais conservador, favorecendo a convergência. Por ser um método de busca local com mecanismo de perturbação controlada, o SA é particularmente útil para refinar soluções em regiões promissoras do espaço de busca.

D. Considerações gerais

A combinação desses três algoritmos oferece um panorama complementar de estratégias de busca: enquanto DE e PSO

são técnicas populacionais capazes de explorar amplamente o espaço de busca, o SA atua como um refinador local com forte capacidade de intensificação. Essa diversidade favorece a identificação de hiperparâmetros robustos para o Transformer simples.

V. Resultados

A. Métricas

A Tabela IV apresenta o desempenho no conjunto de teste para o baseline e para a melhor solução obtida por cada meta-heurística.

Tabela IV
MÉTRICAS DE DESEMPENHO NO CONJUNTO DE TESTE

Método	Acc.	Prec.	Recall	F1
Baseline	0,7700	0,8630	0,6364	0,7326
DE	**0,8600**	0,8517	**0,8918**	**0,8713**
PSO	0,8275	**0,8827**	0,7788	0,8275
SA	0,8413	0,8706	0,8235	0,8464

Todos os algoritmos superaram o baseline em acurácia e F1-score, o que indica que, mesmo sob um orçamento de apenas 12 avaliações, a otimização meta-heurística é capaz de encontrar combinações de hiperparâmetros significativamente melhores do que uma configuração manual.

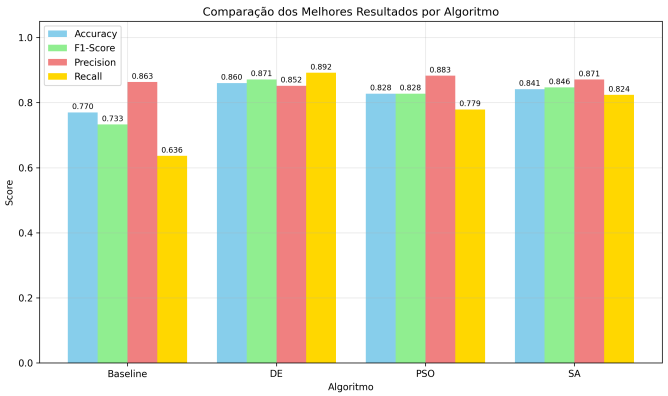

Figura 2. Comparação de métricas entre o baseline e os modelos otimizados.

B. Comparativo de desempenho

A Tabela IV resume os valores absolutos de acurácia, precisão, *recall* e F1-score para o baseline e para cada meta-heurística. Para tornar mais claro o efeito da otimização, a Tabela V apresenta os ganhos absolutos em relação ao baseline (isto é, Δ = método − baseline).

Tabela V
GANHO EM RELAÇÃO AO BASELINE EM CADA MÉTRICA

Método	ΔAcc	ΔPrec.	ΔRecall	ΔF1
DE	+0,0900	-0,0113	+0,2554	+0,1387
PSO	+0,0575	+0,0197	+0,1424	+0,0949
SA	+0,0713	+0,0076	+0,1871	+0,1138

Observa-se que o Differential Evolution (DE) é a abordagem com maior ganho global: ele eleva a acurácia em 0,0900 (9,0 pontos percentuais), o F1-score em 0,1387 (13,9 pontos percentuais) e o *recall* da classe positiva em 0,2554 (25,5 pontos percentuais), ao custo de uma pequena redução na precisão.

O PSO também supera o baseline, com destaque para o aumento da precisão em 0,0197 (2,0 pontos percentuais) e ganhos moderados em acurácia, *recall* e F1-score, configurando um comportamento mais conservador quanto a falsos positivos.

O SA oferece um compromisso entre DE e PSO, apresentando ganhos consistentes em todas as métricas, com aumentos de 0,0713 em acurácia, 0,1138 em F1-score e 0,1871 em *recall*, mantendo a precisão levemente acima do baseline.

De forma geral, o DE é mais indicado quando a prioridade é maximizar a detecção de exemplos positivos (altos ganhos em *recall* e F1-score), enquanto o PSO é atrativo em cenários nos quais falsos positivos são mais críticos. O SA ocupa uma posição intermediária, oferecendo melhorias equilibradas entre precisão e *recall*.

C. Análise de convergência

A seguir são apresentados, separadamente, os gráficos de convergência dos algoritmos DE, PSO e SA em termos de diferentes métricas de desempenho. Em todos os casos, o PSO foi executado com 12 iterações (avaliações de configurações), enquanto DE e SA seguiram o mesmo orçamento de avaliações.

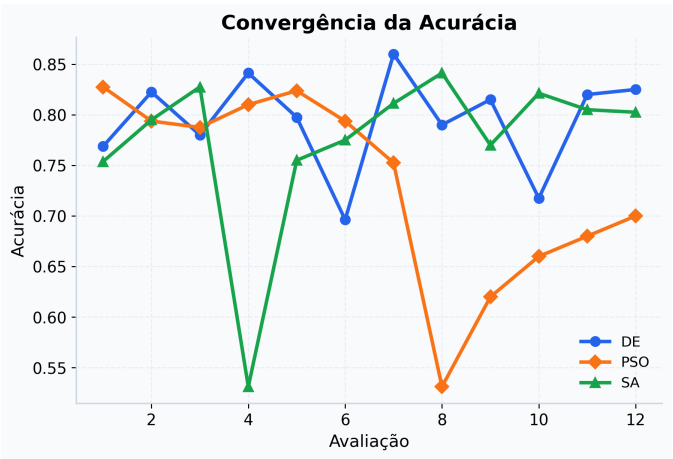

Figura 3. Convergência da acurácia de validação ao longo das avaliações

a) Acurácia de validação.: A Figura 3 mostra a evolução da melhor acurácia de validação obtida por cada algoritmo ao longo das avaliações. Observa-se que o DE atinge rapidamente níveis elevados de acurácia e tende a estabilizar em torno de uma região de alto desempenho, o que sugere uma fase inicial de exploração mais agressiva seguida de intensificação em regiões promissoras do espaço de hiperparâmetros. O PSO apresenta um aumento mais gradual, típico de enxames que vão ajustando as partículas à medida que a informação global

é propagada, enquanto o SA exibe flutuações mais acentuadas nas primeiras avaliações, reflexo da aceitação de soluções piores em temperaturas mais altas.

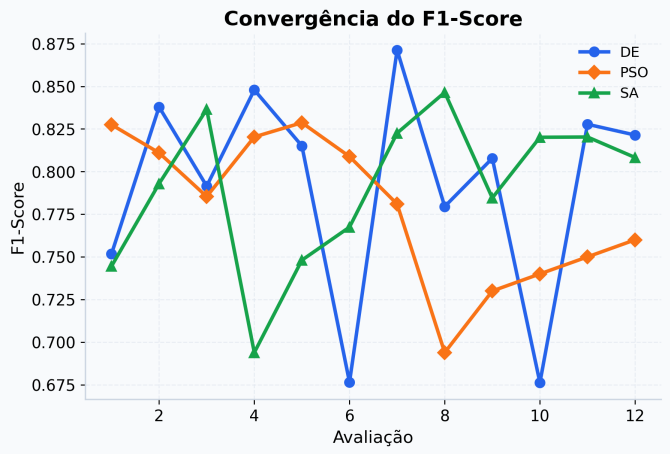

Figura 4. Convergência do F1-score de validação ao longo das avaliações.

b) F1-score de validação.: Na Figura 4 é apresentada a convergência do F1-score de validação. O comportamento geral acompanha o observado para a acurácia, com o DE alcançando rapidamente valores elevados e relativamente estáveis, enquanto o PSO melhora de forma mais progressiva ao longo das iterações. O SA mantém maior variabilidade entre avaliações, o que indica uma exploração mais ampla do espaço de busca, porém com menor consistência na qualidade das soluções intermediárias.

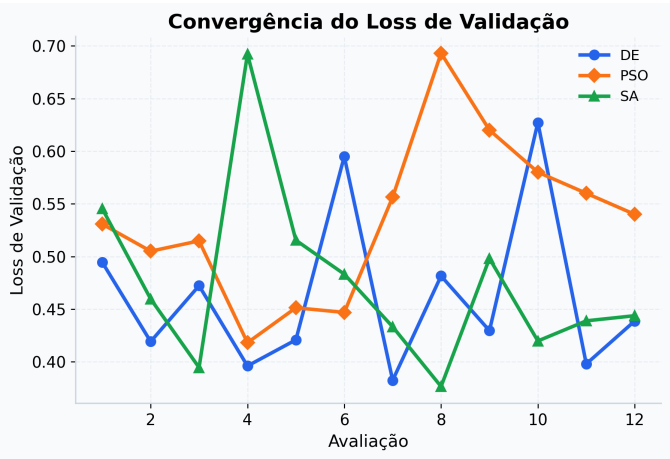

Figura 5. Convergência da função de perda de validação ao longo das avaliações.

c) Perda de validação.: A Figura 5 mostra a evolução da perda de validação. Em linha com as métricas de acurácia e F1-score, o DE apresenta uma redução rápida da perda nas primeiras avaliações, seguida de uma região de estabilização, indicando que o algoritmo consegue encontrar configurações que minimizam a função objetivo com poucas iterações. O PSO promove uma queda mais suave, sugerindo um processo de refinamento incremental, enquanto o SA exibe oscilações mais pronunciadas, coerentes com sua natureza estocástica e com o mecanismo de aceitação de soluções piores durante o resfriamento.

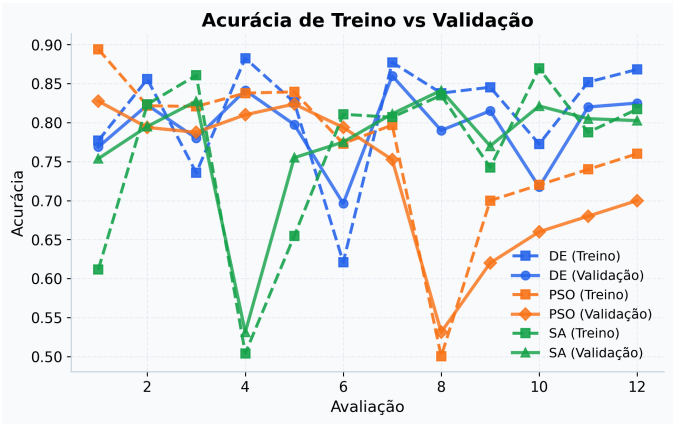

Figura 6. Acurácia de treino e de validação ao longo das avaliações.

d) Acurácia de treino vs. validação.: Por fim, a Figura 6 compara a acurácia de treino e de validação ao longo das avaliações. As curvas de treino e validação para o DE e o PSO permanecem relativamente próximas, sem abertura sustentada entre elas, o que indica ausência de sobreajuste severo dentro do orçamento de avaliações considerado. No caso do SA, observa-se maior oscilação, mas ainda com uma discrepância moderada entre treino e validação, sugerindo que a variabilidade está mais associada ao processo de busca do que a um sobreajuste sistemático. Em conjunto, esses resultados indicam que os algoritmos são capazes de explorar o espaço de hiperparâmetros de forma eficaz sem comprometer a capacidade de generalização do modelo.

VI. Reprodutibilidade e Aspectos Éticos

O código-fonte correspondente ao *pipeline* descrito neste artigo, incluindo implementação do modelo, rotina de treinamento e integração com meta-heurísticas, encontra-se disponível em repositório público [20], com documentação das principais dependências e scripts de execução.

Do ponto de vista ético, a detecção automática de sinais de depressão em redes sociais deve ser encarada como ferramenta de apoio, e não como substituto de avaliação clínica. Falsos negativos podem deixar de sinalizar indivíduos em risco, enquanto falsos positivos podem gerar preocupação desnecessária ou estigma. A coleta e o uso de dados exigem cuidado com privacidade, consentimento, anonimização e possíveis vieses linguísticos e culturais.

VII. Conclusão

Este trabalho investigou o uso de meta-heurísticas para otimização de hiperparâmetros de um Transformer leve em um cenário de orçamento restrito de avaliações da função objetivo. Foram comparados três algoritmos de natureza distinta, *Differential Evolution* (DE), *Particle Swarm Optimization* (PSO) e

Simulated Annealing (SA), aplicados à detecção de sinais de depressão em tweets, tomando como referência um modelo baseline configurado manualmente.

Partindo de uma configuração inicial com acurácia de 0,7700 e F1-score de 0,7326, a melhor solução encontrada pelo DE elevou esses valores para 0,8600 e 0,8713, respectivamente. Além dessas métricas globais, a análise de curvas de convergência e da perda de validação mostrou que o DE atinge rapidamente uma região de alto desempenho e a mantém de forma estável, enquanto PSO e SA exibem dinâmicas de busca mais graduais ou mais ruidosas, mas ainda assim capazes de superar o baseline. Em conjunto, os resultados indicam que, mesmo com apenas 12 avaliações por algoritmo, as meta-heurísticas foram capazes de explorar o espaço de hiperparâmetros de forma mais eficaz do que a configuração manual, melhorando simultaneamente desempenho e estabilidade do Transformer.

Do ponto de vista metodológico, o estudo reforça a viabilidade de empregar meta-heurísticas como camada de otimização em *pipelines* de PLN baseados em Transformers, sobretudo em cenários em que o custo de treinamento impede o uso de buscas exaustivas ou de estratégias clássicas como *grid search* e *random search* em larga escala.

Como trabalhos futuros, pretende-se:

- incorporar validação cruzada estratificada para reduzir a variância das estimativas de desempenho;
- avaliar o *pipeline* em outras bases de dados e tarefas de PLN, de modo a dissociar ainda mais a metodologia do caso específico de detecção de depressão;
- explorar meta-heurísticas adicionais, como algoritmos genéticos e CMA-ES, bem como esquemas híbridos que combinem diferentes estratégias de busca;
- integrar Transformers pré-treinados de maior porte, mantendo a camada meta-heurística como componente central de otimização de hiperparâmetros e avaliando o impacto do aumento de capacidade do modelo sob restrições de orçamento computacional.

REFERÊNCIAS

[1] A. H. Orabi, P. Buddhitha, M. H. Orabi, and D. Inkpen, "Deep learning for depression detection of twitter users," in *Proceedings of the Fifth Workshop on Computational Linguistics and Clinical Psychology*, 2018, pp. 88–97.

[2] A. Qasim *et al.*, "Detection of depression severity in social media text using transformer-based models," *Information*, vol. 16, no. 2, p. 114, 2025.

[3] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, pp. 281–305, 2012.

[4] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in Neural Information Processing Systems 25*, 2012, pp. 2951–2959.

[5] L. Li *et al.*, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *Journal of Machine Learning Research*, vol. 18, no. 185, pp. 1–52, 2018.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30 (NeurIPS 2017)*, 2017, pp. 5998–6008.

[7] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac *et al.*, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 2020, pp. 38–45.

[8] S. Roy *et al.*, "Hyperparameter optimization for deep neural network models: A comprehensive study on methods and techniques," *Innovations in Systems and Software Engineering*, vol. 21, pp. 789–800, 2025.

[9] M. Kaveh *et al.*, "Application of meta-heuristic algorithms for training neural networks: A comprehensive review," *Applied Soft Computing*, vol. 122, p. 108873, 2022.

[10] J. Han *et al.*, "Using differential evolution to improve predictive accuracy of deep learning models applied to pig production data," *Journal of Animal Science*, vol. 98, no. Suppl 3, p. 27, 2020.

[11] A. A. El-Demerdash *et al.*, "Metaheuristic-driven hyperparameter optimization for BERT in sentiment analysis," *Archives for Technical Sciences*, vol. 33, no. 2, pp. 176–192, 2025.

[12] B. Zeng *et al.*, "Particle swarm optimization-based nlp methods for optimizing automatic document classification and retrieval," *PLOS ONE*, vol. 20, no. 7, p. e0325851, 2025.

[13] R. Storn and K. Price, "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[14] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks (ICNN)*, vol. 4, 1995, pp. 1942–1948.

[15] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[16] D. S. Khafaga *et al.*, "Deep learning for depression detection using twitter data," *Intelligent Automation & Soft Computing*, vol. 36, no. 2, pp. 1301–1313, 2023.

[17] R. Kancharapu and S. N. Ayyagari, "Depression detection: Unveiling mental health insights with twitter data and BERT models," *International Journal of Education and Management Engineering*, vol. 14, no. 4, pp. 1–14, 2024.

[18] P. R. Lorenzo *et al.*, "Particle swarm optimization for hyper-parameter selection in deep neural networks," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, 2017, pp. 481–488.

[19] InFamousCoder, "Depression: Twitter dataset + feature extraction," https://www.kaggle.com/datasets/infamouscoder/mental-health-social-media, 2022, acessado em nov. 2025.

[20] B. L. e Lucas Pimenta, "Otimização de hiperparâmetros de modelos transformer via meta-heurísticas de otimização: Uma aplicação em detecção de depressão em tweets," 2025, repositório público. Disponível em: https://github.com/lucaspimentab/evo-transformer-hparam-tuning.