# **🐍 Guia de Instalação e Execução - API Reptile**

## **📋 Visão Geral**

Esta API Flask é um sistema de classificação de texto usando modelos PyTorch (.pt) para categorizar textos em 10 categorias diferentes. O sistema foi projetado para trabalhar com modelos LLM salvos no formato PyTorch.

## **🔧 Pré-requisitos**

### **Versões Recomendadas**

* Python 3.8 ou superior
* Sistema operacional: Windows, Linux ou macOS
* Pelo menos 4GB de RAM livre
* Espaço em disco: 2GB para dependências

### **Ferramentas Necessárias**

* Git (opcional, para controle de versão)
* Editor de código (VS Code, PyCharm, etc.)
* Terminal/Prompt de comando

## **📁 Estrutura de Pastas Necessária**

Antes de começar, crie a seguinte estrutura de pastas:

seu-projeto/

├── app.py # Arquivo principal da API (seu código)

├── models/ # Pasta para o modelo

│ └── llama-pth-checkpoint/

│ └── melhor\_modelo\_pt.pt # Seu modelo treinado

├── uploads/ # Pasta para uploads (criada automaticamente)

├── requirements.txt # Dependências (será criado)

└── README.md # Documentação

## **🚀 Passo a Passo para Instalação**

### **Passo 1: Preparar o Ambiente**

#### **1.1 Criar pasta do projeto**

mkdir api-reptile

cd api-reptile

#### **1.2 Criar ambiente virtual (OBRIGATÓRIO)**

# Windows

python -m venv venv

venv\Scripts\activate

# Linux/macOS

python3 -m venv venv

source venv/bin/activate

**⚠️ IMPORTANTE**: Sempre ative o ambiente virtual antes de instalar pacotes!

### **Passo 2: Instalar Dependências**

#### **2.1 Criar arquivo requirements.txt**

Crie um arquivo chamado requirements.txt com o seguinte conteúdo:

Flask==2.3.3

Flask-CORS==4.0.0

torch==2.0.1

pandas==2.0.3

numpy==1.24.3

Werkzeug==2.3.7

transformers==4.30.0

**⚠️ IMPORTANTE**: A versão BERT requer a biblioteca transformers para o tokenizer!

#### **2.2 Instalar pacotes**

pip install -r requirements.txt

**Alternativa manual**:

pip install Flask Flask-CORS torch pandas numpy Werkzeug transformers

**⚠️ ATENÇÃO**: Se você tiver problemas com a instalação do transformers, tente:

pip install transformers --no-cache-dir

### **Passo 3: Preparar o Modelo**

#### **3.1 Criar estrutura de pastas**

mkdir -p models/llama-pth-checkpoint

#### **3.2 Colocar o modelo**

* Copie seu arquivo melhor\_modelo\_pt.pt para models/llama-pth-checkpoint/
* Certifique-se de que o caminho está correto: models/llama-pth-checkpoint/melhor\_modelo\_pt.pt

### **Passo 4: Configurar o Código BERT**

#### **4.1 Salvar o código corrigido**

* **IMPORTANTE**: Use o código BERT corrigido (não o código original com features manuais)
* Salve como app.py na pasta raiz do projeto

#### **4.2 Primeira execução (Download do BERT)**

Na primeira execução, o sistema baixará automaticamente:

* **Modelo BERT**: neuralmind/bert-base-portuguese-cased (~400MB)
* **Tokenizer**: Vocabulário e configurações

# Primeira execução pode demorar para baixar o BERT

python app.py

**⚠️ IMPORTANTE**: Certifique-se de ter conexão com internet estável!

### **Passo 5: Testar a Instalação**

#### **5.1 Verificar dependências BERT**

python -c "import flask, torch, pandas, numpy, transformers; print('Todas as dependências BERT instaladas com sucesso!')"

#### **5.2 Testar o tokenizer BERT**

python -c "from transformers import BertTokenizer; tokenizer = BertTokenizer.from\_pretrained('neuralmind/bert-base-portuguese-cased'); print('Tokenizer BERT funcionando!')"

#### **5.2 Verificar o modelo**

python -c "import os; print('Modelo encontrado:', os.path.exists('models/llama-pth-checkpoint/melhor\_modelo\_pt.pt'))"

## **▶️ Executando a API**

### **Método 1: Execução Direta**

python app.py

### **Método 2: Usando Flask (alternativo)**

export FLASK\_APP=app.py # Linux/macOS

set FLASK\_APP=app.py # Windows

flask run --host=0.0.0.0 --port=5000

### **🎯 Verificando se Funcionou**

Se tudo estiver correto, você verá uma mensagem similar a:

🐍 API REPTILE CORRIGIDA - v2.0\_fixed

Status: ✅ OK

Modelo: ✅ Carregado

Categorias: 10 disponíveis

...

\* Running on all addresses (0.0.0.0)

\* Running on http://127.0.0.1:5000

\* Running on http://[seu-ip]:5000

## **🧪 Testando a API**

### **Teste 1: Health Check**

curl http://localhost:5000/health

### **Teste 2: Categorias Disponíveis**

curl http://localhost:5000/categorias

### **Teste 3: Classificação de Texto**

curl -X POST http://localhost:5000/classify \

-H "Content-Type: application/json" \

-d '{"text": "Como você está se sentindo hoje?"}'

### **Teste 4: Teste Forçado (IMPORTANTE)**

curl -X POST http://localhost:5000/force-test \

-H "Content-Type: application/json" \

-d '{}'

## **🔧 Endpoints da API BERT**

| **Método** | **Endpoint** | **Descrição** | **Novo/Modificado** |
| --- | --- | --- | --- |
| GET | / | Informações básicas da API BERT | ✅ Atualizado |
| GET | /health | Status BERT + Tokenizer | ✅ Atualizado |
| GET | /categorias | Lista de categorias terapêuticas | ✅ Atualizado |
| POST | /classify | Classifica texto (com contexto opcional) | 🆕 **Novo formato** |
| POST | /classify-batch | Múltiplos textos ou interações | 🆕 **Novo formato** |
| POST | /test-tokenizer | Testa tokenização BERT | 🆕 **Novo** |
| POST | /force-test-bert | Teste forçado categorias específicas | 🆕 **Novo** |
| POST | /processar | CSV com Fala\_Terapeuta + Fala\_Cliente | ✅ Atualizado |

### **📋 Novos Formatos de Requisição**

#### **Classificação Individual:**

{

"text": "Como você está se sentindo hoje?",

"contexto\_cliente": "Estou ansioso" // Opcional

}

#### **Lote Simples:**

{

"texts": [

"Como você está?",

"Entendo sua dificuldade"

]

}

#### **Lote com Contexto:**

{

"interacoes": [

{

"texto": "Como você está?",

"contexto\_cliente": "Estou ansioso"

}

]

}

## **❌ Problemas Comuns BERT e Soluções**

### **Problema 1: "Tokenizer não carregado"**

**Causa**: Biblioteca transformers não instalada ou modelo BERT não baixado **Solução**:

# Reinstalar transformers

pip uninstall transformers

pip install transformers --no-cache-dir

# Verificar conexão de internet

python -c "from transformers import BertTokenizer; BertTokenizer.from\_pretrained('neuralmind/bert-base-portuguese-cased')"

### **Problema 2: "HTTP 404 - neuralmind/bert-base-portuguese-cased"**

**Causa**: Problema de conectividade com HuggingFace **Solução**:

# Usar proxy ou VPN se necessário

export HF\_ENDPOINT=https://huggingface.co

# Download manual

python -c "

from transformers import BertTokenizer, BertForSequenceClassification

tokenizer = BertTokenizer.from\_pretrained('neuralmind/bert-base-portuguese-cased')

model = BertForSequenceClassification.from\_pretrained('neuralmind/bert-base-portuguese-cased', num\_labels=10)

print('Download concluído!')

"

### **Problema 3: "BERT sempre retorna mesma categoria"**

**Causa**: Modelo .pt não é compatível com BERT **Solução**:

1. Execute o teste forçado: POST /force-test-bert
2. Verifique se o modelo foi treinado com BERT
3. **CRÍTICO**: O modelo deve ter sido salvo após treinamento com model.state\_dict()

### **Problema 4: "OutOfMemoryError"**

**Causa**: BERT consome muita RAM **Solução**:

# Reduzir batch size no código ou usar CPU

export CUDA\_VISIBLE\_DEVICES="" # Forçar CPU

# Ou adicionar no código:

device = torch.device("cpu") # Sempre usar CPU

### **Problema 5: "Erro ao carregar state\_dict"**

**Causa**: Incompatibilidade entre modelo treinado e arquitetura **Solução**:

# Verificar as chaves do modelo salvo

import torch

checkpoint = torch.load('models/llama-pth-checkpoint/melhor\_modelo\_pt.pt', map\_location='cpu')

print("Tipo:", type(checkpoint))

if isinstance(checkpoint, dict):

print("Chaves:", list(checkpoint.keys())[:10])

## **🔍 Diagnóstico Avançado**

### **Verificar Logs**

# Executar com logs detalhados

python app.py 2>&1 | tee app.log

### **Testar Modelo Manualmente**

import torch

# Carregar modelo

model = torch.load('models/llama-pth-checkpoint/melhor\_modelo\_pt.pt', map\_location='cpu')

print(f"Tipo do modelo: {type(model)}")

print(f"Chaves (se dict): {list(model.keys()) if isinstance(model, dict) else 'Não é dict'}")

### **Verificar Features**

# Testar extração de features

from app import extract\_simple\_features

features = extract\_simple\_features("Texto de teste", target\_size=3072)

print(f"Shape: {features.shape}")

print(f"Valores únicos: {len(torch.unique(features))}")

## **📝 Uso em Produção**

### **Configurações Recomendadas**

# Substituir na linha final do app.py

if \_\_name\_\_ == '\_\_main\_\_':

app.run(

debug=False, # Desabilitar debug

host='0.0.0.0', # Aceitar conexões externas

port=5000, # Porta padrão

threaded=True # Permitir múltiplas conexões

)

### **Usando Gunicorn (Recomendado)**

pip install gunicorn

gunicorn --bind 0.0.0.0:5000 --workers 4 app:app

## **🔐 Segurança**

### **Configurações de Segurança**

* Desabilite CORS em produção (remova origins=["\*"])
* Use HTTPS em produção
* Implemente autenticação se necessário
* Limite o tamanho dos uploads

### **Exemplo de Configuração Segura**

CORS(app, origins=["http://localhost:3000", "https://seudorminio.com"])

app.config['MAX\_CONTENT\_LENGTH'] = 16 \* 1024 \* 1024 # 16MB max

## **📊 Monitoramento**

### **Logs de Produção**

import logging

logging.basicConfig(

level=logging.INFO,

format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',

handlers=[

logging.FileHandler('api.log'),

logging.StreamHandler()

]

)

## **🆘 Suporte**

### **Checklist de Verificação**

* [ ] Python 3.8+ instalado
* [ ] Ambiente virtual ativado
* [ ] Dependências instaladas
* [ ] Modelo na pasta correta
* [ ] Permissões corretas
* [ ] Porta 5000 livre
* [ ] Teste forçado executado

### **Comandos de Debug**

# Verificar versão Python

python --version

# Verificar pacotes instalados

pip list

# Verificar estrutura de pastas

find . -type f -name "\*.pt"

# Verificar logs

tail -f app.log

## **🎉 Pronto para Usar!**

Se seguiu todos os passos, sua API BERT deve estar funcionando em:

* **URL Local**: http://localhost:5000
* **Health Check**: http://localhost:5000/health
* **Interface HTML**: Use o arquivo teste\_api\_bert.html fornecido

### **✅ Checklist Final BERT**

* [ ] Python 3.8+ instalado
* [ ] Ambiente virtual ativado
* [ ] Dependências instaladas (incluindo transformers)
* [ ] Modelo .pt na pasta correta
* [ ] Primeira execução completada (BERT baixado)
* [ ] Teste de health retorna tokenizer\_loaded: true
* [ ] Teste BERT forçado executado com sucesso

### **🎯 Próximos Passos BERT:**

1. **Execute o teste BERT forçado** (POST /force-test-bert) - CRÍTICO
2. **Teste com contexto** - use cliente + terapeuta
3. **Processe CSVs** com colunas Fala\_Terapeuta e Fala\_Cliente
4. **Use a interface HTML** para testes visuais
5. **Configure para produção** se aplicável

### **🚨 Se Algo Não Funcionar:**

* Verifique os logs para mensagens de erro
* Execute todos os testes da seção "Testando a API BERT"
* Confirme que o modelo foi treinado com BERT (não features manuais)
* Verifique se há conexão com internet (para download do BERT)

**Lembre-se**: Esta versão usa **BERT português** e é **incompatível** com modelos treinados com features manuais!

## **🔄 Migração da Versão Antiga**

Se você tinha a versão com features manuais:

1. **Backup** do código antigo
2. **Substitua** completamente pelo código BERT
3. **Retreine** o modelo usando o script de treinamento BERT
4. **Teste** a nova versão

**Não é possível** usar modelos antigos com a nova API BERT!

Boa sorte com sua API BERT! 🚀🧠