Sistema de Identificação de Dados Sensíveis

1º Hackathon em Controle Social: Desafio Participa DF

Categoria: Acesso à Informação

Autores:

- Vinicius Armando Menezes de Andrade
- João Luiz de Jesus Amaro

Data: Janeiro/2026

Sumário

---

1. Descrição do Projeto

Este sistema foi desenvolvido para identificar automaticamente **dados pessoais sensíveis** em pedidos de acesso à informação, auxiliando órgãos públicos na proteção de informações pessoais conforme a **Lei Geral de Proteção de Dados (LGPD)**.

O sistema utiliza uma **abordagem híbrida** que combina três técnicas complementares para maximizar a precisão da detecção:

- **Regex + Validadores**: Validação estrutural de documentos brasileiros (CPF, RG, CNPJ)
- **GLiNER (Machine Learning)**: Modelo de NER especializado em PII para detecção contextual
- **LLM (Opcional)**: Gemini API como camada de fallback para casos subjetivos

Características Principais

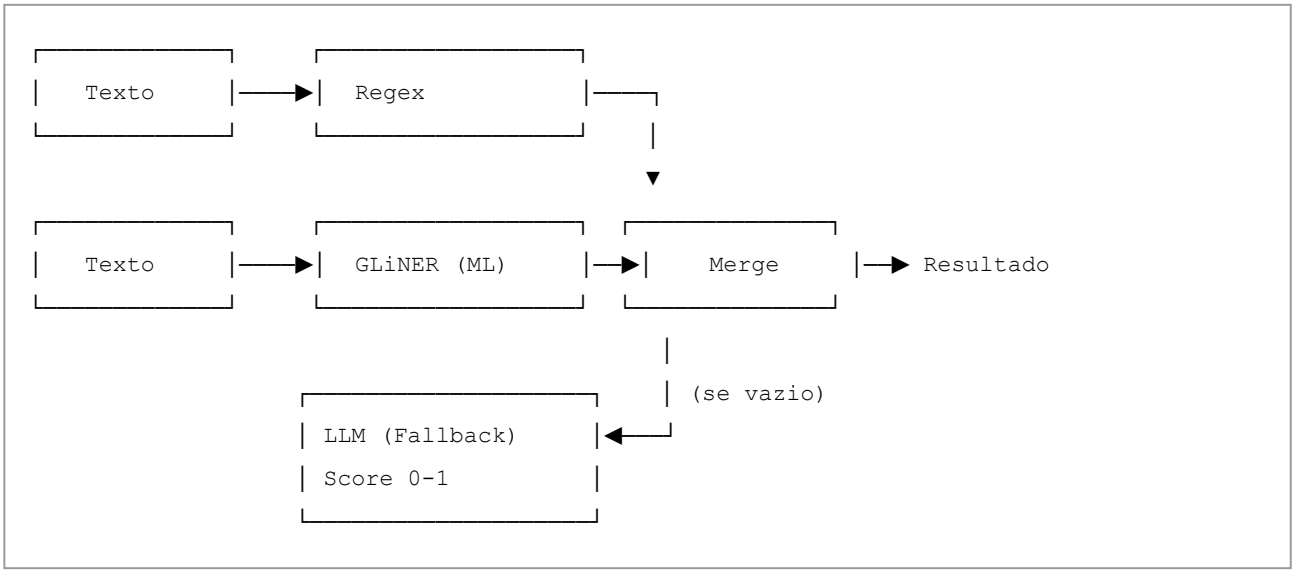
Característica	Descrição
□ Funciona sem API	Sistema independente de quotas ou chaves de API
□ Alta Precisão	Combina múltiplas técnicas de detecção
□ Formato Padrão	Saída CSV no formato `ID,Predicao` (0 ou 1)
□ Multi-formato	Suporta Excel, CSV, JSON, Parquet, etc.
□ Auto-detecção	Identifica automaticamente colunas de ID e texto

---

2. Arquitetura do Sistema

O sistema utiliza uma arquitetura em camadas que processa o texto sequencialmente, combinando os resultados das diferentes técnicas de detecção.

Fluxo de Processamento



Estrutura de Arquivos

```
sistema_identificacao_dados_sensiveis/
├── main.py                    # Script principal de execução
├── predict.py                 # Script de predição para submissão
├── cli.py                     # Interface de linha de comando interativa
├── requirements.txt           # Dependências do projeto
├── Dockerfile                 # Configuração Docker
├── docker-compose.yml         # Orquestração de containers
├── src/
│   ├── __init__.py           # Módulo principal
│   ├── detectores.py         # Classes de detecção (híbrido)
│   ├── detector_gliner.py    # Detector GLiNER (ML)
│   ├── carregador.py         # Carregamento de dados multi-formato
│   └── metricas.py           # Métricas e geração de relatórios
├── dados/
│   ├── nomes_proprios.json   # Lista de nomes próprios brasileiros
│   └── sobrenomes.json       # Lista de sobrenomes comuns
└── resultados/                # Diretório para saídas
```

---

3. Tipos de Dados Detectados

O sistema identifica diversos tipos de dados pessoais sensíveis:

Tipo	Descrição	Método de Detecção
Nome	Identificação de pessoa natural	GLiNER + Dicionário
CPF	Cadastro de Pessoa Física	GLiNER + Regex + Validação
CNPJ	Cadastro Nacional da Pessoa Jurídica	GLiNER + Regex
RG	Registro Geral	Regex + Contexto
Telefone	Números de contato	GLiNER + Regex + DDD
E-mail	Endereço de correio eletrônico	GLiNER + Regex
Endereço/CEP	Localização física	GLiNER + Regex

Modelo GLiNER

O sistema utiliza o modelo `E3-JSI/gliner-multi-pii-domains-v1` (https://huggingface.co/E3-JSI/gliner-multi-pii-domains-v1):

- **50+ tipos de PII** suportados
- **Suporte a português** e dados brasileiros
- **GPU (CUDA)** para inferência rápida
- **Lazy loading** para economia de memória

---

4. Instalação e Configuração

Pré-requisitos

- **Python**: 3.9 ou superior
- **GPU (Opcional)**: CUDA compatível para aceleração do GLiNER

Instalação Padrão

```
# Criar ambiente virtual
python -m venv venv
source venv/bin/activate  # Linux/macOS
# ou: venv\Scripts\activate  # Windows

# Instalar dependências
pip install -r requirements.txt
```

Dependências Principais

Pacote	Versão	Descrição
`pandas`	≥1.5.0	Manipulação de dados
`openpyxl`	≥3.0.0	Leitura de arquivos Excel
`gliner`	≥0.2.0	Modelo de NER para PII
`torch`	≥2.0.0	Backend de ML

Instalação com Docker (Alternativa)

```
# Build da imagem
docker-compose build

# Execução com CLI interativa
docker-compose run --rm acesso-info python cli.py

# Execução com predict.py
docker-compose run --rm acesso-info python predict.py dados_entrada/arquivo.xlsx saida.csv
```

Configuração Opcional da API (LLM)

```
# Definir API key (opcional)
export GEMINI_API_KEY='sua-chave-aqui'

# Ou criar arquivo .env
echo "GEMINI_API_KEY=sua_chave_aqui" > .env
```

---

5. Como Usar

Script de Predição (Para Submissão)

```
# Uso básico
python predict.py <arquivo_entrada> <arquivo_saida>

# Exemplos
python predict.py dados_teste.xlsx predicoes.csv
python predict.py pedidos.csv resultado.csv

# Especificando coluna de texto manualmente (opcional)
python predict.py dados.xlsx saida.csv "Texto do Pedido"
```

Formato de Entrada

O sistema aceita diversos formatos de arquivo:

Formato	Extensões
Excel	`.xlsx`, `.xls`
CSV	`.csv`
TSV	`.tsv`
JSON	`.json`
Texto	`.txt`
Parquet	`.parquet`

O arquivo deve conter:

- **Coluna ID**: Identificador único do registro
- **Coluna de Texto**: Texto a ser analisado (auto-detectado)

Formato de Saída

O sistema gera um arquivo CSV padrão:

```
ID,Predicao
1,0
2,1
3,0
```

Valor	Significado
`0`	NÃO contém dados pessoais (pode ser público)
`1`	Contém dados pessoais (NÃO deve ser público)

Uso como Biblioteca Python

```
from src.detectores import SistemaDeteccaoIntegrado

# Inicialização padrão
sistema = SistemaDeteccaoIntegrado()

# Analisar texto
texto = "Meu nome é João Silva, CPF 123.456.789-09"
resultado = sistema.obter_resumo(texto)

print(f"Contém dados: {resultado['contem_dados_pessoais']}")
print(f"Tipos detectados: {resultado['por_tipo']}")

# Verificação simples
if sistema.contem_dados_pessoais(texto):
    print("□ NÃO publicar - contém dados pessoais")
else:
    print("□ Pode publicar")
```

---

6. Métricas de Avaliação

O sistema calcula as métricas conforme especificado no edital:

Métrica	Fórmula	Descrição
Precisão	`VP / (VP + FP)`	Dos detectados, quantos realmente continham dados
Recall	`VP / (VP + FN)`	Dos que continham, quantos foram detectados
F1-Score	`2 × (P × R) / (P + R)`	Média harmônica entre Precisão e Recall

Legenda

- **VP (Verdadeiro Positivo)**: Corretamente identificado como sensível
- **VN (Verdadeiro Negativo)**: Corretamente identificado como público
- **FP (Falso Positivo)**: Incorretamente identificado como sensível
- **FN (Falso Negativo)**: Dado sensível não identificado

---

7. Considerações Finais

Limitações

Limitação	Descrição
Primeira execução	Download automático do modelo GLiNER (~1.2GB)

Limitação	Descrição
Hardware	GLiNER funciona melhor com GPU (CUDA)
Nomes estrangeiros	Podem ter menor taxa de detecção
API Gemini	Free tier tem limites baixos (15 req/min)

Diferenciais do Sistema

- **Abordagem híbrida**: Combina Regex, ML e LLM para máxima precisão
- **Independência**: Funciona sem conexão com APIs externas
- **Flexibilidade**: Suporta múltiplos formatos de entrada
- **Escalabilidade**: Preparado para processamento em lote
- **Código aberto**: Transparência e possibilidade de auditoria

Repositório

 GitHub: github.com/vini-andra/Hackathon-CGDF-Acesso-Info (https://github.com/vini-andra/Hackathon-CGDF-Acesso-Info)

1º Hackathon em Controle Social do Distrito Federal

Categoria: Acesso à Informação

Janeiro/2026