```
"""
Ollama 統合異常検知システム（メインプログラム）
Ollama（ローカルLLM）を使用したZero-shot/Few-shot異常検知
CSV形式のデータを読み込んで分析
"""
import traceback
import csv
from typing import List, Dict
import os
import argparse
import sys
import httpx

# OpenAI APIクライアントをインポート
try:
    import openai
except ImportError:
    print("⚠  OpenAI SDKがインストールされていません。")
    print("   以下のコマンドでインストールしてください:")
    print("   pip install openai")
    sys.exit(1)

# Few-shotサンプルをインポート
try:
    from few_shot_examples import get_few_shot_examples, load_custom_few_shot
except ImportError:
    print("⚠  few_shot_examples.py が見つかりません。")
    print("   Few-shot検知を使用する場合は、few_shot_examples.py を同じディレクトリに配置してください。")

    def get_few_shot_examples(data_type: str) -> str:
        return ""

    def load_custom_few_shot(filepath: str) -> str:
        return ""


class OpenAIAnomalyDetector:
    """OpenAI APIを使用した異常検知クラス"""

    def __init__(self, runtime:str , model_name: str , base_url: str , api_key: str = None):
        self.model_name = model_name
        self.base_url = base_url
        self.api_endpoint = f"{base_url}/v1"
        # TIME OUT
        timeout=httpx.Timeout(30.0, read=20.0, write=5.0, connect=5.0)

        if runtime == "OPENAI":
            self.base_url="OpenAI"
            self.model_name = model_name
            if not api_key:
                raise ValueError(
                    "OpenAI APIキーが設定されていません。¥n"
                    "以下のいずれかの方法で設定してください:¥n"
                    "  1. 環境変数: export OPENAI_API_KEY='your-api-key'¥n"
                    "  2. 引数指定: python anomaly_detection.py --api-key YOUR_API_KEY"
                )
            self.client = openai.OpenAI(
                api_key=api_key,
                timeout=timeout
            )
        else:
            self.client = openai.OpenAI(
                base_url=self.api_endpoint,
                api_key=api_key,
                timeout=timeout
            )

            # OpenAIクライアントの初期化

    def query_llm(self, prompt: str) -> str:
        """OpenAI APIにクエリを送信"""
        Temperature=0  # 回答のランダム性 この値を 大きくすると 回答の変化が大きい
        MaxTokens=2000
        try:
            response = self.client.chat.completions.create(
                model=self.model_name,
                messages=[
                    {"role": "system", "content": "あなたは経験豊富なエンジニアです。"},
                    {"role": "user", "content": prompt}
                ],
                temperature=Temperature,
                max_tokens=MaxTokens
            )
            return response.choices[0].message.content
        except openai.APITimeoutError as e:
            print(e, flush=True)
        except Exception as e:
            print(traceback.format_exc())
```

```
            return f"エラー: {str(e)}"

    def detect_network_anomaly(self, traffic_data: List[Dict], mode: str = "zero-shot",
                               custom_few_shot: str = None) -> Dict:
        """ネットワークトラフィック異常検知"""

        if mode == "few-shot":
            if custom_few_shot:
                few_shot_examples = custom_few_shot
            else:
                few_shot_examples = get_few_shot_examples('network')
        else:
            few_shot_examples = ""

        traffic_summary = "¥n".join([
            f"- [{i+1}] 送信元IP: {t['src_ip']}, 宛先IP: {t['dst_ip']}, ポート: {t['port']}, "
            f"プロトコル: {t['protocol']}, パケット数: {t['packets']}, バイト数: {t['bytes']}"
            for i, t in enumerate(traffic_data)
        ])

        prompt = f"""あなたはネットワークセキュリティの専門家です。以下のネットワークトラフィックログを分析し、異常を検出してください。

{few_shot_examples}

# 分析対象のトラフィック:
{traffic_summary}

各トラフィックについて:
1. 正常 or 異常 を判定
2. 異常と判定した場合、その理由を説明
3. リスクレベル（低/中/高）を評価

簡潔に回答してください。
"""

        response = self.query_llm(prompt)
        return {"raw_response": response, "traffic_data": traffic_data}

    def detect_log_anomaly(self, log_entries: List[Dict], mode: str = "zero-shot",
                           custom_few_shot: str = None) -> Dict:
        """システム・アプリログの異常検知"""

        if mode == "few-shot":
            if custom_few_shot:
                few_shot_examples = custom_few_shot
            else:
                few_shot_examples = get_few_shot_examples('log')
        else:
            few_shot_examples = ""

        log_summary = "¥n".join([
            f"- [{i+1}] [{l['timestamp']}] {l['level']}: {l['message']}"
            for i, l in enumerate(log_entries)
        ])

        prompt = f"""あなたはシステムセキュリティの専門家です。以下のシステムログを分析し、異常を検出してください。

{few_shot_examples}

# 分析対象のログ:
{log_summary}

各ログエントリについて:
1. 正常 or 異常 を判定
2. 異常と判定した場合、その理由と推奨対応を説明
3. 緊急度（低/中/高/緊急）を評価

簡潔に回答してください。
"""

        response = self.query_llm(prompt)
        return {"raw_response": response, "log_entries": log_entries}

    def detect_sensor_anomaly(self, sensor_data: List[Dict], mode: str = "zero-shot",
                              custom_few_shot: str = None) -> Dict:
        """センサ時系列データの異常検知"""

        if mode == "few-shot":
            if custom_few_shot:
                few_shot_examples = custom_few_shot
            else:
                few_shot_examples = get_few_shot_examples('sensor')
        else:
            few_shot_examples = ""

        sensor_summary = "¥n".join([
```

```
            f"- [{i+1}] 時刻: {s['timestamp']}, 温度: {s['temperature']}℃, 振動: {s['vibration']}mm/s"
            for i, s in enumerate(sensor_data)
        ])

        prompt = f"""あなたは産業機器の保守エンジニアです。以下のセンサー時系列データを分析し、異常を検出してください。

{few_shot_examples}

# 分析対象のセンサーデータ:
{sensor_summary}

データ全体を分析し:
1. 正常 or 異常 を判定
2. 異常なパターン・トレンドがあれば説明
3. 予測される故障モードと推奨メンテナンス時期
4. リスクレベル（正常/注意/警告/危険）を評価

簡潔に回答してください。
"""

        response = self.query_llm(prompt)
        return {"raw_response": response, "sensor_data": sensor_data}


# ======================================
# CSV データ読み込み関数
# ======================================

def load_csv_data(filename: str) -> List[Dict]:
    """CSVファイルからデータを読み込み"""
    try:
        data = []
        with open(filename, 'r', encoding='utf-8') as f:
            reader = csv.DictReader(f)
            for row in reader:
                data.append(row)

        print(f"✓ データ読み込み成功: {filename} ({len(data)}件)")
        return data
    except FileNotFoundError:
        print(f"✗ ファイルが見つかりません: {filename}")
        print(f"  'python generate_sample_data.py' を実行してサンプルデータを生成してください。")
        return []
    except Exception as e:
        print(f"✗ データ読み込みエラー: {e}")
        return []


# ======================================
# メイン実行部分
# ======================================

def main():
    """メイン実行関数"""
    parser = argparse.ArgumentParser(
        description='OpenAI API異常検知システム',
        formatter_class=argparse.RawDescriptionHelpFormatter,
        epilog="""
使用例:
  # 環境変数でAPIキーを設定して実行
  ## Linux
  export OPENAI_API_KEY='your-api-key-here'
  ## WIndows
  $Env:OPENAI_API_KEY = 'your-api-key-here'
  python anomaly_detection.py

  # APIキーを引数で指定
  python anomaly_detection.py --api-key YOUR_API_KEY

  # Zero-shot検知を使用
  python anomaly_detection.py --mode zero-shot

  # 分析件数を指定
  python anomaly_detection.py --limit 20

  # カスタムデータディレクトリを指定
  python anomaly_detection.py --data-dir ./my_data

  # すべてのデータを分析（制限なし）
  python anomaly_detection.py --limit -1

  # 特定のデータタイプのみ分析
  python anomaly_detection.py --only network
  python anomaly_detection.py --only logs
  python anomaly_detection.py --only sensor

  # カスタムFew-shotファイルを使用
  python anomaly_detection.py --mode few-shot --few-shot-network my_network_examples.txt
```

```
# 別PCのLLM (OLLAMA)を使用
python anomaly_detection.py --url http://192.168.10.36:11434 --model llama3.2-vision:11b
"""
  )
  # 基本設定
  parser.add_argument('--runtime', type=str, default='Ollama',
                      help='使用するLLMランタイム (デフォルト: Ollama) ')
  parser.add_argument('--model', type=str, default='NONE',
                      help='使用するOllamaモデル (デフォルト: llama3.2:3b ) ')
  parser.add_argument('--url', type=str, default='http://localhost:11434',
                      help='OllamaサーバーのURL (デフォルト: http://localhost:11434) ')

  parser.add_argument('--mode', type=str, choices=['zero-shot', 'few-shot'],
                      default='few-shot',
                      help='検知モード (デフォルト: few-shot) ')

  # データ設定
  parser.add_argument('--data-dir', type=str, default='sample_data',
                      help='データディレクトリのパス (デフォルト: sample_data) ')
  parser.add_argument('--limit', type=int, default=10,
                      help='分析するデータ件数の上限 (-1で全件、デフォルト: 10) ')

  # 分析対象の選択
  parser.add_argument('--only', type=str,
                      choices=['network', 'logs', 'sensor', 'all'],
                      default='all',
                      help='分析するデータタイプ (デフォルト: all) ')

# カスタムFew-shotファイル
parser.add_argument('--few-shot-network', type=str, default=None,
                    help='ネットワークトラフィック用カスタムFew-shotファイル')
parser.add_argument('--few-shot-log', type=str, default=None,
                    help='システムログ用カスタムFew-shotファイル')
parser.add_argument('--few-shot-sensor', type=str, default=None,
                    help='センサーデータ用カスタムFew-shotファイル')

# 引数をパース
args = parser.parse_args()
if args.runtime == "Ollama" :
    api_key = "NONE"
    if args.model == "NONE":
        model_name = "llama3.2:3b"
    else:
        model_name = args.model
else :
    # APIキーの取得 (引数 > 環境変数)
    if args.api_key is None:
        api_key = os.environ.get("OPENAI_API_KEY")
    else:
        api_key = args.api_key

    if args.model == "NONE":
        model_name = "gpt-4.1-nano"
    else:
        model_name = args.model

print("=" * 70)
print("LLM 異常検知システム")
print("=" * 70)
print()

# 検知器の初期化
print("LLM 異常検知システムを初期化中...")
try:
    detector = OpenAIAnomalyDetector(runtime=args.runtime , model_name=model_name,base_url=args.url, api_key=api_key)
    print(f"✓ 接続先: {detector.base_url}")
    print(f"✓ 使用モデル: {detector.model_name}")
    print(f"✓ APIキー: {'設定済み' if api_key else '未設定'}")
    print(f"✓ 検知モード: {args.mode}")
    print(f"✓ 分析件数制限: {'全件' if args.limit == -1 else f'{args.limit}件'}")
    print(f"✓ 対象データ: {args.only}")
except Exception as e:
    print(f"✗ 初期化エラー: {e}")
    sys.exit(1)
print()

# CSVファイルパス
data_dir = args.data_dir
network_file = os.path.join(data_dir, "network_traffic.csv")
logs_file = os.path.join(data_dir, "system_logs.csv")
sensor_file = os.path.join(data_dir, "sensor_data.csv")

# ファイル存在確認
missing_files = []
if args.only in ['network', 'all'] and not os.path.exists(network_file):
    missing_files.append(network_file)
if args.only in ['logs', 'all'] and not os.path.exists(logs_file):
```

```
    missing_files.append(logs_file)
if args.only in ['sensor', 'all'] and not os.path.exists(sensor_file):
    missing_files.append(sensor_file)

if missing_files:
    print("⚠  以下のサンプルデータファイルが見つかりません:")
    for f in missing_files:
        print(f"   · {f}")
    print()
    print("以下のコマンドでサンプルデータを生成してください:")
    print("   python generate_sample_data.py")
    print()
    sys.exit(1)

# 1. ネットワークトラフィック異常検知
if args.only in ['network', 'all']:
    print("¥n" + "=" * 70)
    print("【1. ネットワークトラフィック異常検知】")
    print("=" * 70)
    network_data = load_csv_data(network_file)
    if network_data:
        # データ件数制限
        if args.limit > 0:
            network_data = network_data[:args.limit]

        # カスタムFew-shotの読み込み
        custom_few_shot = None
        if args.mode == 'few-shot' and args.few_shot_network:
            custom_few_shot = load_custom_few_shot(args.few_shot_network)
            if custom_few_shot:
                print(f"✓ カスタムFew-shotを読み込みました: {args.few_shot_network}")

        print(f"¥n分析対象: {len(network_data)}件のトラフィック")
        print(f"¥n[{args.mode}検知実行中...]")
        result = detector.detect_network_anomaly(network_data, mode=args.mode,
                                                 custom_few_shot=custom_few_shot)
        print(f"¥nLLM分析結果:¥n{result['raw_response']}")

# 2. システムログ異常検知
if args.only in ['logs', 'all']:
    print("¥n" + "=" * 70)
    print("【2. システムログ異常検知】")
    print("=" * 70)

    log_data = load_csv_data(logs_file)
    if log_data:
        # データ件数制限
        if args.limit > 0:
            log_data = log_data[:args.limit]

        # カスタムFew-shotの読み込み
        custom_few_shot = None
        if args.mode == 'few-shot' and args.few_shot_log:
            custom_few_shot = load_custom_few_shot(args.few_shot_log)
            if custom_few_shot:
                print(f"✓ カスタムFew-shotを読み込みました: {args.few_shot_log}")

        print(f"¥n分析対象: {len(log_data)}件のログエントリ")
        print(f"¥n[{args.mode}検知実行中...]")
        result = detector.detect_log_anomaly(log_data, mode=args.mode,
                                             custom_few_shot=custom_few_shot)
        print(f"¥nLLM分析結果:¥n{result['raw_response']}")

# 3. センサー時系列異常検知
if args.only in ['sensor', 'all']:
    print("¥n" + "=" * 70)
    print("【3. センサー時系列データ異常検知】")
    print("=" * 70)

    sensor_data = load_csv_data(sensor_file)
    if sensor_data:
        # センサーデータは全体を分析（時系列の文脈が重要）
        # ただしlimitが指定されている場合は制限
        if args.limit > 0 and len(sensor_data) > args.limit:
            print(f"   ※ センサーデータは時系列なので最新の{args.limit}件を分析します")
            sensor_data = sensor_data[-args.limit:]

        # カスタムFew-shotの読み込み
        custom_few_shot = None
        if args.mode == 'few-shot' and args.few_shot_sensor:
            custom_few_shot = load_custom_few_shot(args.few_shot_sensor)
            if custom_few_shot:
                print(f"✓ カスタムFew-shotを読み込みました: {args.few_shot_sensor}")

        print(f"¥n分析対象: {len(sensor_data)}データポイント")
        print(f"¥n[{args.mode}検知実行中...]")
        result = detector.detect_sensor_anomaly(sensor_data, mode=args.mode,
                                                custom_few_shot=custom_few_shot)
```

```
            print(f"¥nLLM分析結果:¥n{result['raw_response']}")

    print("¥n" + "=" * 70)
    print("分析完了")
    print("=" * 70)
    print()
    print("【オプション例】")
    print("  ・Zero-shot検知: python anomaly_detection.py --mode zero-shot")
    print("  ・別モデル使用: python anomaly_detection.py --model llama3.1:8b")
    print("  ・分析件数変更: python anomaly_detection.py --limit 50")
    print("  ・全件分析: python anomaly_detection.py --limit -1")
    print("  ・カスタムFew-shot: python anomaly_detection.py --few-shot-network my_examples.txt")
    print("  ・ヘルプ表示: python anomaly_detection.py --help")


if __name__ == "__main__":
    main()
```