IndiaAI CyberGuard AI Hackathon

Crime Classification Using Fine-tuned T5 Model

Team Members

- Santhosh G S - santhoshgs013@gmail.com
- Omkar Tupe Tanaji - omkartupe84@gmail.com

Project Overview

This project aims to develop a machine learning solution for classifying cybersecurity crimes into categories and sub-categories based on detailed crime descriptions. The pipeline uses natural language processing techniques and implements a fine-tuned T5 model to handle this multi-label classification task.

Problem Statement

The challenge involves creating an automated system that can:

- Analyze detailed crime descriptions
- Classify crimes into main categories
- Further categorize them into specific sub-categories
- Deal with cases where crime description may not be well detailed

Dataset Description

The dataset consists of crime reports with three main components:

- **Crime Additional Information**: Detailed textual descriptions of crimes
- **Category**: Main crime classification
- **Sub-category**: Specific classification within the main category

Initial dataset sizes:

- Training set: 93,686 records
- Test set: 31,229 records

Technical Approach

1. Data Analysis and Preprocessing

1. **Missing Value Analysis**
    - Identified and handled missing crime descriptions
    - Discovered patterns in missing sub-categories, particularly for sensitive crimes
    - Implemented strategic imputation for missing sub-categories
2. **Label Analysis**
    - 15 main categories and 35 original sub-categories
    - Post-imputation: 39 sub-categories
    - Identified hierarchical relationships between categories and sub-categories
3. **Data Cleaning**
    - Removed duplicate entries
    - Special handling for cases with multiple classifications

2. Model Development and Training

Model Architecture

- **Base Model**: Flan-T5-small (google/flan-t5-small)
- **Architecture Type**: Encoder-Decoder Transformer (80M params)
- **Approach**: Fine-tuning for crime classification

Input Processing

1. **Prompt Engineering**
    - Structured prompt template incorporating:
        - Crime description
        - Available categories and sub-categories
    - Allowed for multi-label classification (up to 3 categories/sub-categories)
    - **Format**: "category: <respective_category> subcategory: <respective_subcategory>"

Training Configuration

1. **Dataset Split**
    - Training set: 95% of preprocessed data
    - Validation set: 5% of preprocessed data
    - Test set: Separate held-out dataset
2. **Training Parameters**
    - Learning rate: 2e-3
    - Batch size: 16 per device
    - Training epochs: 3
    - Weight decay: 0.01
    - Optimization features:
        - Mixed precision training (FP16)
        - Gradient accumulation
3. **Training Monitoring**

- Custom logging callback implementation
- TensorBoard integration
- Metrics tracking:
  - Training loss
  - Validation loss
  - Resource utilization

Model Output

- Generates both category and subcategory predictions
- Handles multi-label cases through ' or ' separated predictions
- Maintains hierarchical relationship between categories and sub-categories

Dependency Details

```
transformers==4.31.0

datasets==2.14.5

pandas==2.0.3

numpy==1.25.2

scikit-learn==1.3.0

torch==2.0.1
```

Project Structure

```
.
├── categories.pkl                # Serialized categories mapping
├── subcategories.pkl             # Serialized subcategories mapping
├── Inferencing_and_Results.ipynb        # Model evaluation notebook
├── Data_Analysis_and_Preprocessing.ipynb # Data preprocessing nb
├── finetuning_fT5.py                      # Model training script
├── requirements.txt                        # Project dependencies
├── logs/
│   └── training_log_fT5.txt               # Training logs
├── dataset/
│   ├── train.csv                          # Original training data
│   └── test.csv                           # Original test data
└── cleaned_dataset/
    ├── cleaned_train_dataset.csv        # Preprocessed training data
    └── cleaned_test_dataset.csv           # Preprocessed test data
```

Compute Resources

- **GPU:** NVIDIA A100 (80GB VRAM)
- **CPU:** AMD EPYC Processor
- **RAM:** 220GB

Model Evaluation and Results

Inference Pipeline

1. **Input Processing**
    - Batch size: 128
    - Maximum sequence length: 1500 tokens
    - GPU-accelerated inference
    - Structured output parsing using regex
2. **Evaluation Metrics** The model was evaluated at three distinct levels:

Item-level Performance (Combined Category and Subcategory)

- Accuracy: 49.14%
- Precision: 11.54%
- Recall: 9.63%
- F1-Score: 9.53%

Category-level Performance

- Accuracy: 73.99%
- Precision: 19.12%
- Recall: 14.65%
- F1-Score: 15.70%

Subcategory-level Performance

- Accuracy: 49.54%
- Precision: 13.24%
- Recall: 10.96%
- F1-Score: 10.96%

Performance Analysis

1. **Main Category Classification**
    - Achieved robust 73.99% accuracy in main category identification

    - Strong performance considering the complexity of cybersecurity incidents
    - Effective handling of overlapping categories
2. **Hierarchical Classification**
    - Consistent performance across subcategory classification (49.54%)
    - Maintained stability in combined item-level classification (49.14%)
    - Successfully preserved category-subcategory relationships
3. **Resource Efficiency**
    - Achieved competitive results using T5-small architecture
    - Optimized for deployment in resource-constrained environments
    - Fast inference time with batch processing

Technical Challenges and Solutions

1. **Multi-label Classification**
    - Challenge: Handling cases where crimes span multiple categories
    - Solution: Implemented "or" separated predictions in the understanding that the description was ambiguous to specifically place them under single category/sub-category
2. **Resource Optimization**
    - Challenge: Processing large text sequences with limited resources
    - Solution: Used a light weight model with batch processing and GPU acceleration
3. **Output Consistency**
    - Challenge: Maintaining structured output format
    - Solution: Implemented regex-based output parsing

Future Improvements

1. **Model Architecture**
    - Experiment with larger T5 variants for potential accuracy improvements
    - Fine Tuning mini LLMs using LoRA on the same pipeline
2. **Training Strategy**
    - Try LoRA fine-tuning strategy
    - Try increasing the training epochs to observe how it impacts the prediction accuracy and model's generalizability
    - Explore curriculum learning by feeding together similar crime types

Installation and Usage

```
# Clone the repository
git clone [repository-url]

# Install dependencies
pip install -r requirements.txt
```

```
# Run preprocessing
Jupyter execute Data_Analysis_and_Preprocessing.ipynb

# Train model
python3 finetuning_fT5.py

# Run inference
Jupyter execute Inferencing_and_Results.ipynb
```

Conclusion

This work represents our initial efforts toward addressing the challenge of classifying cybersecurity complaints. Moving forward, we aim to delve into more advanced techniques and methodologies to enhance accuracy, scalability, and robustness in solving this critical problem.