HW3 experiment Report

Name:李崇楷 Student ID: 313834006

Code: https://github.com/kailee0422/RNN-Transformer/tree/main/HW3

Introduction

This project focuses on building a Named Entity Recognition (NER) tool using the BERT model. I train and evaluate the model on the DNRTI dataset, following standard procedures with train.txt, valid.txt, and test.txt. To further enhance performance, Ialso explore the use of SecBERT with BiLSTM and CRF. The goal is to understand and implement a modern NER pipeline based on transformer architectures.

Method

This section details the implementation of our SecBERT–BiLSTM–CRF model for Named Entity Recognition (NER), covering data preparation, model design, embedding integration, training procedure, and evaluation.

Data Preprocessing

- Baseline tokenization : For the baseline model, I use standard tokenization (e.g., spaCy or WordPiece from a generic BERT), replace digits with zero, lowercase all tokens, and pad/truncate sequences to a fixed length. No domain-specific normalization is applied.
- SecBERT tokenization : In addition to the baseline steps, I employ the SecBERT tokenizer with its security-optimized vocabulary. This tokenizer better handles security jargon and multi-word expressions common in vulnerability descriptions.
- CoNLL-style Input: I read raw files where each line contains a token and its IOB tag, sentences separated by blank lines. Malformed lines and lines with only one token are ignored.
- Token–Tag Extraction: For each sentence, we collect tokens and their corresponding tags into Python lists.
- Tag Mappings: We build tag2idx and idx2tag dictionaries by enumerating the sorted set of all tags across train/valid/test splits.
- Sequence Encoding: Using jackaduma/SecBERT tokenizer, we convert each sentence to subword token IDs (input_ids) and attention masks, padded/truncated to a fixed length (256).
- Label Alignment: We align original word-level tags to subword positions, assigning the tag only to the first subword of each word and using the 'O' tag for padding or special tokens..

Model Architecture:SecBERT–BiLSTM–CNN-CRF

My model begins by encoding each sentence with SecBERT (jackaduma/SecBERT), producing 768-dimensional contextualized embeddings with 0.1 dropout. A two-layer bidirectional LSTM (with 0.2 dropout) then transforms these embeddings into 512-dimensional token features (256 per direction). For character-level modeling when using CNN mode, we embed each character (25-dim), apply a 1×3 convolution and max-pool across characters to yield a 25-dim vector, which is concatenated to the BiLSTM output. Finally, a linear layer projects the combined features to the tag space, and a CRF layer models tag transitions, optimizing negative log-likelihood during training and decoding via Viterbi at inference.

Difference between Baseline and SecBERT

Baseline Embedding: the model uses either randomly initialized word embeddings or generic pre-trained embeddings (e.g. GloVe or standard BERT) without security tuning.

SecBERT Embedding: SecBERT is a BERT model further pre-trained on security-related corpora, producing contextual representations that capture domain-specific semantics (e.g. vulnerabilities, attack patterns).

The integration code remains identical; only the encoder weights differ, enabling direct ablation.

Training Setup

Baseline BiLSTM–CRF Training

- Optimizer: Stochastic Gradient Descent (SGD) with momentum = 0.9 and initial learning rate = 0.015.
- Learning-rate Decay: At the end of each epoch, decay LR by factor 1/(1 + decay_rate * epoch) where decay_rate=0.05.
- Gradient Clipping: Clip gradients to a maximum norm of 5.0 to prevent explosion.
- Batching & Updates: Online updates (batch size = 1 sentence) shuffled each epoch; evaluate on dev set after each full pass.
- Epochs: Up to 50 epochs, saving best checkpoint by dev F1; early stopping not applied explicitly.

SecBERT–BiLSTM-CNN–CRF Training

- Optimizer: AdamW with weight decay (1e-4) and initial learning rate = 1e-4, better suited for fine-tuning large pre-trained Transformers.
- Scheduler: ReduceLROnPlateau monitoring validation F1; reduce LR by factor 0.5 if no improvement for 2 consecutive epochs.
- Gradient Clipping: Not applied (empirically unnecessary when fine-tuning SecBERT with AdamW).
- Mini-batch Training: Batch size = 8 sentences for efficient GPU utilization
- Epochs & Checkpointing: 10 epochs; save model checkpoint with highest validation F1.
- Validation Frequency: Evaluate on dev set at end of each epoch.

By differentiating optimizers, batch strategies, and learning-rate schedules, we tailor training to the characteristics of non-pretrained BiLSTM vs. large Transformer fine-tuning.

Evaluation

We report both entity-level and overall performance:

Entity-level Precision: For each named-entity category (e.g., PER, LOC, ORG), we compute the precision of predicted spans.

Overall Metrics: We calculate micro-averaged precision, recall, and F1-score across all entity types to assess general model performance.

Result

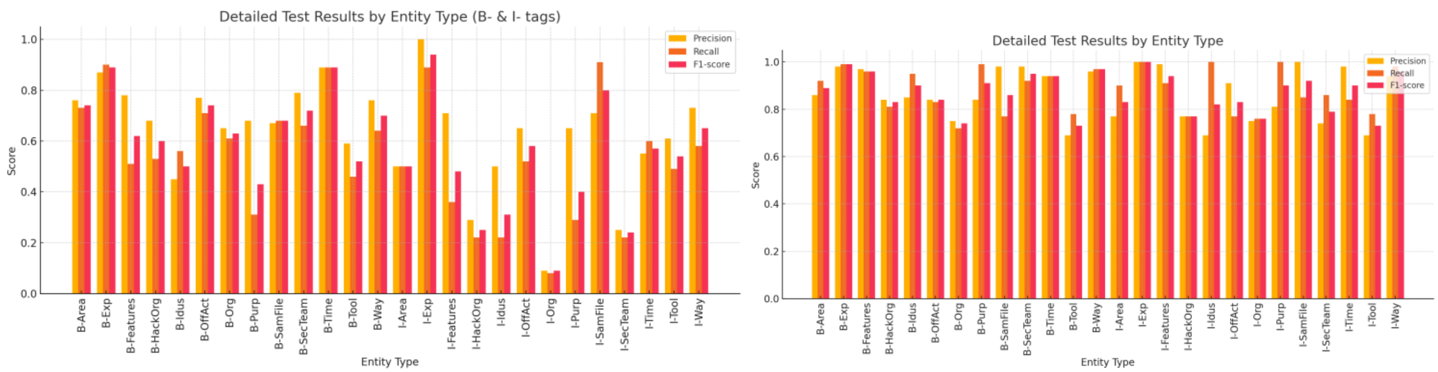

Figure 1 . Baseline(left) and After using SecBERT(right) B- and I-tag score comparison.

Entity	Precision	Recall	F1-score
B-Area	0.76	0.73	0.74
B-Exp	0.87	0.9	0.89
B-Features	0.78	0.51	0.62
B-HackOrg	0.68	0.53	0.6
B-Idus	0.45	0.56	0.5
B-OffAct	0.77	0.71	0.74
B-Org	0.65	0.61	0.63
B-Purp	0.68	0.31	0.43
B-SamFile	0.67	0.68	0.68
B-SecTeam	0.79	0.66	0.72
B-Time	0.89	0.89	0.89
B-Tool	0.59	0.46	0.52
B-Way	0.76	0.64	0.7
I-Area	0.5	0.5	0.5
I-Exp	1.0	0.89	0.94
I-Features	0.71	0.36	0.48
I-HackOrg	0.29	0.22	0.25
I-Idus	0.5	0.22	0.31
I-OffAct	0.65	0.52	0.58
I-Org	0.09	0.08	0.09
I-Purp	0.65	0.29	0.4
I-SamFile	0.71	0.91	0.8
I-SecTeam	0.25	0.22	0.24
I-Time	0.55	0.6	0.57
I-Tool	0.61	0.49	0.54
I-Way	0.73	0.58	0.65

Entity	Precision	Recall	F1-score
B-Area	0.86	0.92	0.89
B-Exp	0.98	0.99	0.99
B-Features	0.97	0.96	0.96
B-HackOrg	0.84	0.81	0.83
B-Idus	0.85	0.95	0.9
B-OffAct	0.84	0.83	0.84
B-Org	0.75	0.72	0.74
B-Purp	0.84	0.99	0.91
B-SamFile	0.98	0.77	0.86
B-SecTeam	0.98	0.92	0.95
B-Time	0.94	0.94	0.94
B-Tool	0.69	0.78	0.73
B-Way	0.96	0.97	0.97
I-Area	0.77	0.9	0.83
I-Exp	1.0	1.0	1.0
I-Features	0.99	0.91	0.94
I-HackOrg	0.77	0.77	0.77
I-Idus	0.69	1.0	0.82
I-OffAct	0.91	0.77	0.83
I-Org	0.75	0.76	0.76
I-Purp	0.81	1.0	0.9
I-SamFile	1.0	0.85	0.92
I-SecTeam	0.74	0.86	0.79
I-Time	0.98	0.84	0.9
I-Tool	0.69	0.78	0.73
I-Way	0.94	0.98	0.96

Figure 2 . Baseline(left) and After using SecBERT(right) B- and I-tag metrics comparison.

	Testing Precision	Testing Recall	Testing F1
Baseline	71.75%	56.52%	64.69%
SecBERT	**82.46%**	**84.50%**	**83.47%**

Table 1. Testing metrics with an Baseline and after using SecBERT for embedding.

Conclusion

We compared a standard BiLSTM–CRF baseline with a SecBERT–BiLSTM–CNN–CRF model on the DNRTI NER task. Incorporating SecBERT raised test F1 from 64.7% to 83.5%, showing that domain-tuned embeddings greatly improve entity recognition in security text. This demonstrates the value of security-focused pre-training for specialized NER applications.