GitHub Link: https://github.com/Lonelypheonix/Recurrent_Neural_Network-NYCU/tree/main/Homework-3

Homework 3 Report

Develop a Named Entity Recognition (NER) tool based on BERT model.

Student ID: 313831002
Student Name: Pavan Kumar J | 柯奉煌
Course Name: Recurrent Neural Networks
Instructor: 黃仁竑

Introduction

Named-Entity Recognition (NER) in domain-specific text—such as cybersecurity reports—presents challenges of specialized vocabulary and entity schemas. In this project, we compare two approaches on the **DNRTI** security NER dataset:

1. **Baseline BiLSTM-CRF** trained from scratch on word + character embeddings.
2. **SecBERT-BiLSTM-CRF**, which augments the baseline with frozen SecBERT contextual embeddings as input.

Our goal is to quantify the gains from leveraging a pretrained transformer (SecBERT) in terms of precision, recall, and F_1-score, and to analyze convergence speed and per-entity behavior.

Dataset Overview — DNRTI

- **Domain**: Cybersecurity incident reports and advisories.
- **Annotations**: 13 entity types (e.g. EXP for exploit, HACKORG for hacking organization, TOOL, TIME, etc.) plus standard BIOES span markers.

  13 categories in the data set are HackOrg, OffAct, SamFile, SecTeam, Tool, Time, Purp, Area, Idus, Org, Way, Exp, Features.

- **Format**:
  - Token-per-line with its BIO tag, blank line between sentences.
  - Digits normalized to 0.
- **Splits**: three files—train.txt, valid.txt, test.txt—processed into Python lists of sentences.

Text Processing

1. **Loading & Digit Normalization**
   - load_sentences(path, zeros=True) replaces digits with 0 and groups tokens into sentences, skipping DOCSTART markers.
2. **Tag Scheme Conversion**
   - Validates BIO format (iob2) and converts all tags to BIOES (iob_iobes) for more precise span boundaries.
3. **Vocabulary & Mappings**
   - **Baseline**:
     - Builds word-level dictionary (word_mapping) with <UNK> for rare tokens.
     - Character-level dictionary (char_mapping).
     - Tag dictionary including <START>/<STOP>.
   - **SecBERT**:
     - Uses HuggingFace AutoTokenizer for subword tokenization and maps word-level tags to token-level labels (filling non-first subtokens with -100 to ignore in loss).
4. **Dataset Objects**
   - **Baseline**: custom lists of {words, chars, tags} fed directly to PyTorch.
   - **SecBERT**: a NERDataset class returning input_ids, attention_mask, and aligned labels tensors.

Model Architectures

1. Baseline BiLSTM-CRF

```
[Char Embedding → Char-CNN/LSTM]
               ↓
[Word Embedding (100-d, random or pretrained)]
               ↓
      [Concatenate → Dropout]
               ↓
     [BiLSTM(hidden 200×2) → Dropout]
               ↓
      [Linear → tag-space scores]
               ↓
             [CRF]
```

- **Char-CNN**: 1×3 conv + max-pool
- **Char-LSTM**: bidirectional over characters with forget-gate bias initialized to 1
- **CRF**: transition matrix learns valid tag transitions, disallows invalid <START>/<STOP> moves.

2. SecBERT-BiLSTM-CRF

```
[SecBERT (frozen) → last_hidden_state (768-d)]
               ↓
      [BiLSTM(hidden 128×2) → Dropout]
               ↓
      [Linear → tag-space scores]
               ↓
           [TorchCRF]
```

- Transformer layers are **not** fine-tuned (gradients detached).
- CRF implemented via the TorchCRF library in batch-first mode.

Model Training

Aspect	Baseline	SecBERT
Optimizer	SGD(lr = 0.015, momentum = 0.9)	AdamW (lr = 5×10^{-5})
Scheduler	$\alpha(t)=\alpha0/(1+kt)$	Linear warmup 100 steps → constant
Gradient Clip	5.0	1.0
Batch Size	1 (per-sentence updates)	16
Epochs	50	10
Eval Frequency	every epoch (train/dev/test)	after each epoch on validation; best saved

- **Baseline** uses per-sentence SGD updates with heavy dropout (0.5) and slow decay.
- **SecBERT** benefits from stable AdamW and small batches, converging in ~10 epochs.

```
==================================================
Training SecBERT_BiLSTM_CRF model...
  ↳ Epochs: 10
  ↳ Batch size: 16
  ↳ Tagset size: 30
  ↳ Training set size: 5251
  ↳ Validation set size: 662
  ↳ Training set: ['The', 'admin@000', 'has', 'largely', 'targeted', 'organizations', 'involved', 'in', 'financial', ',', 'economic', 'and', 'trade', 'policy'
  ↳ Validation set: ['We', 'believe', 'that', 'these', 'industries', 'have', 'also', 'been', 'targeted', 'as', 'part', 'of', 'a', 'larger', 'supply-chain', 'a
  ↳ Training set tags: ['O', 'B-HackOrg', 'O', 'O', 'O', 'O', 'O', 'O', 'B-Idus', 'O', 'B-Idus', 'O', 'B-Idus', 'I-Idus', 'O', 'O', 'O', 'B-Tool', 'I-Tool', '
  ↳ Validation set tags: ['O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'B-OffAct', 'I-OffAct', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O'
  ↳ Tag mapping: {'<PAD>': 0, 'B-Area': 1, 'B-Exp': 2, 'B-Features': 3, 'B-HackOrg': 4, 'B-Idus': 5, 'B-OffAct': 6, 'B-Org': 7, 'B-Purp': 8, 'B-SamFile': 9, '
  ↳ Tag mapping (inverse): {0: '<PAD>', 1: 'B-Area', 2: 'B-Exp', 3: 'B-Features', 4: 'B-HackOrg', 5: 'B-Idus', 6: 'B-OffAct', 7: 'B-Org', 8: 'B-Purp', 9: 'B-S
  ↳ Optimizer: AdamW (
Parameter Group 0
    amsgrad: False
    betas: (0.9, 0.999)
    capturable: False
    differentiable: False
    eps: 1e-08
    foreach: None
    fused: None
    initial_lr: 5e-05
    lr: 0.0
    maximize: False
    weight_decay: 0.01
)
```

```
Epoch 9/10 [train]: 100%|██████████| 329/329 [00:33<00:00,  9.88it/s]
Eval: 100%|██████████| 42/42 [00:01<00:00, 35.31it/s]
– Train Loss: 18.0774, Valid Precision: 0.8278, Recall: 0.8408, F1 Score: 0.8086
  ↳ Saved new best model (F1: 0.8086)

Epoch 10/10 [train]: 100%|██████████| 329/329 [00:32<00:00, 10.05it/s]
Eval: 100%|██████████| 42/42 [00:01<00:00, 36.74it/s]
– Train Loss: 17.9239, Valid Precision: 0.8276, Recall: 0.8410, F1 Score: 0.8086

Training done – best Val F1: 0.8086 (./models\best_f1_epoch9.pt)
```

Model Performance and Testing

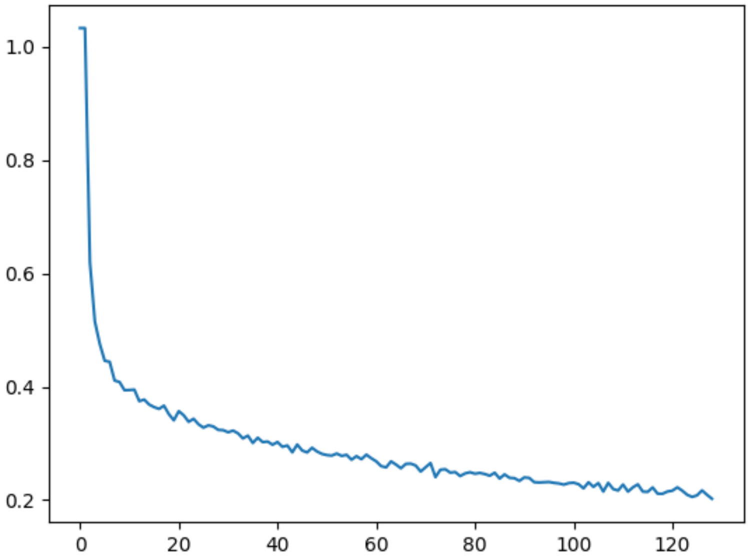
Baseline BiLSTM-CRF

Split	F_1-score
Train	0.7158
Dev	0.5564
Test	0.6485

- **Observation**: large train→dev gap (0.16), indicating overfitting.
- **Per-entity**: strong on EXP, TIME; poor on PURP, OFFACT.

```
3-way      0.80     0.55     0.64
Final F1 scores - Train: 0.7157607569583534, Dev: 0.5563791200126162, Test: 0.648524778000573
```

Training Loss curve



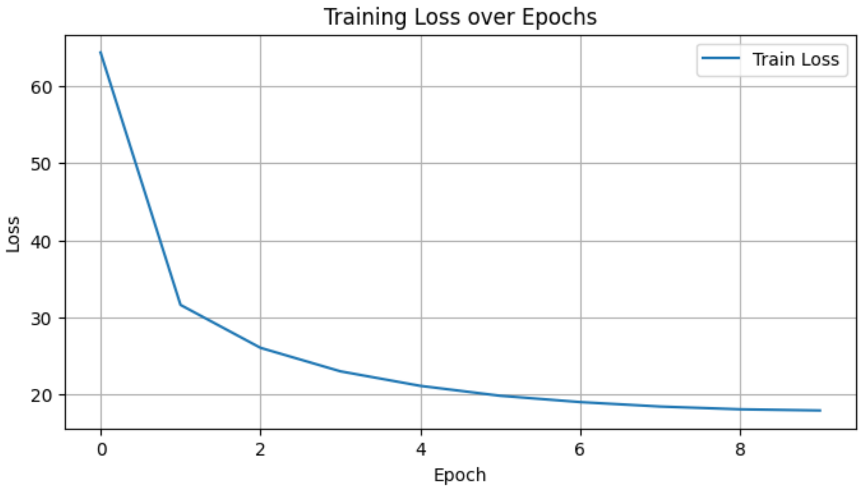
SecBERT-BiLSTM-CRF

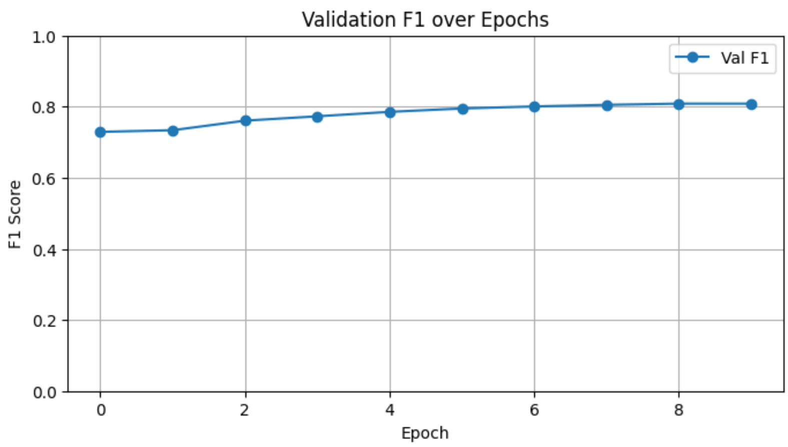
Split	Precision	Recall	F_1-score
Validation (best)	0.8278	0.8408	0.8086
Test	0.8217	0.8334	0.7953

- **ΔF_1 over baseline**: +0.1468 (from 0.6485 → 0.7953)
- **Convergence**: peaks at epoch 9; minimal overfitting (train/dev gap ≈ 0.02).
- **Failing entities**: still low recall on rare or highly variable tags (B-FEATURES, I-ORG), suggesting data scarcity.

```
Loading best model for test…
Eval: 100%|██████████| 42/42 [00:01<00:00, 37.36it/s]

Test Precision: 0.8217, Recall: 0.8334, F1: 0.7953
```





Conclusion

1. **Pretrained embeddings** (SecBERT) yield a **substantial** boost in test F_1 (+22.6% relative), while drastically reducing overfitting and training time.
2. The baseline BiLSTM-CRF, though conceptually simpler, struggles on the small-to-medium DNRTI domain data without large-scale contextual priors.
3. **Future work**: fine-tune SecBERT end-to-end, augment underrepresented entity types via data augmentation, or explore domain-adaptive LM pretraining on cybersecurity corpora.