Recurrent Neural Networks
Homework #3

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

1. Introduction

This homework aims to build a Named Entity Recognition (NER) system specifically for cybersecurity-related texts using the DNTRI dataset. In this assignment, I completed the following steps:

- Built a hybrid NER model combining SecBERT, BiLSTM, and CRF for contextualized sequence labeling.
- Preprocessed DNTRI's train.txt, valid.txt, and test.txt into sentences and aligned labels.
- Used the SecBERT tokenizer for word-piece encoding and label alignment. SecBERT was pre-trained on cybersecurity-related corpora, making it more suitable for the DNTRI dataset than generic BERT.
- Implemented a custom PyTorch Dataset and DataLoader for batched training.
- Trained and validated the model using dynamic learning rate scheduling and early stopping
- Visualized training loss, validation loss, and F1 score trends.
- Evaluated test performance using seqeval metrics and heatmap visualization.

2. Models and Implementation

2.1 Overall Implementation Flow

1. Load and parse train.txt, valid.txt, and test.txt from the DNTRI dataset.
2. Use SecBERT tokenizer to encode sentences into subword tokens.
3. Align word-level labels with subword-level tokens and pad them appropriately.
4. Create a PyTorch Dataset and DataLoader to feed the model.
5. Define the SecBERT_BiLSTM_CRF model combining SecBERT, a BiLSTM, and a CRF.
6. Train the model with Adam optimizer, validation on each epoch, early stopping, and dynamic learning rate adjustment using ReduceLROnPlateau.
7. Evaluate the final model on test data and generate classification heatmap.

2.2 Data Preprocessing

Each input file (train.txt, valid.txt, test.txt) contains one word per line followed by its entity label. Sentences are separated by blank lines. I parse this into lists of words and corresponding labels, resulting in train_sentences, train_labels, etc.

2.3 Tokenizer

I use BertTokenizerFast from Hugging Face's transformers library, specifically loading the jackaduma/SecBERT model. Tokenization is performed with is_split_into_words=True and includes offset mapping to align word labels with tokens. Padding and truncation are applied with a max sequence length of 128.

2.4 Model: SecBERT_BiLSTM_CRF

- SecBERT Encoder: Generates contextual embeddings for each token.
- BiLSTM Layer: A bidirectional LSTM with hidden size 256 captures forward and backward sequential context.
- Linear Layer: Maps LSTM outputs to emission scores for each label.
- CRF Layer: Enforces sequence-level consistency in output labels.
- Dropout: Optional dropout can be added after LSTM for regularization (not yet shown to improve performance significantly).

The BiLSTM is used to capture bidirectional dependencies across tokens, while the CRF layer models label transitions explicitly, which is particularly beneficial for enforcing valid label sequences in NER tasks.

3. Training and Evaluation Strategy

Component	Setting
Batch Size	8
Optimizer	Adam
Learning rate	5e-5
Loss Function	Negative Log-Likelihood from CRF
Epochs	50 with early stopping (best at 40)
Scheduler	ReduceLROnPlateau (mode='max', patience=2)
Dropout	Optional (p=0.1 tested, not used in final)

Evaluation Metrics	Precision, Recall, F1-score
Device	GPU

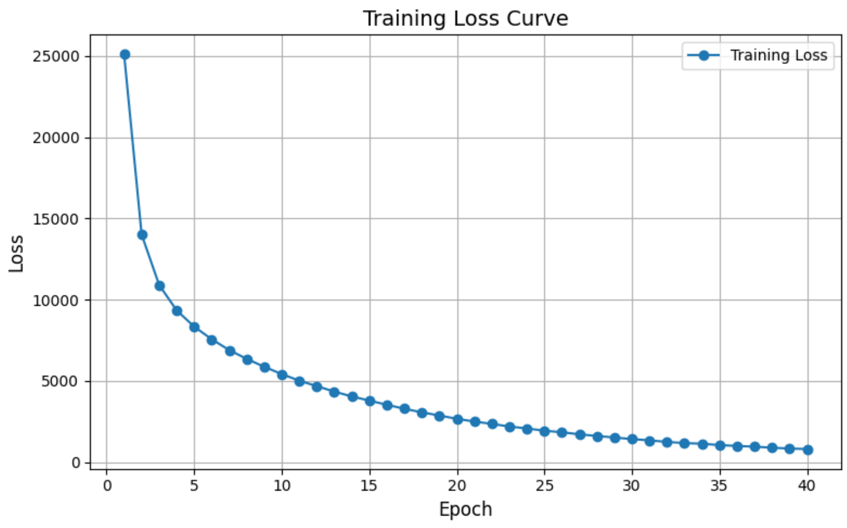
I train the model using the Adam optimizer with a learning rate of 5e-5. Validation F1 score is used as the metric for learning rate scheduling and early stopping.Training stops if no improvement is observed for 3 consecutive epochs. Each epoch reports training loss, validation loss, and F1 score.

After training, I evaluate the model on the test set and generate detailed classification metrics using heatmap visualization.

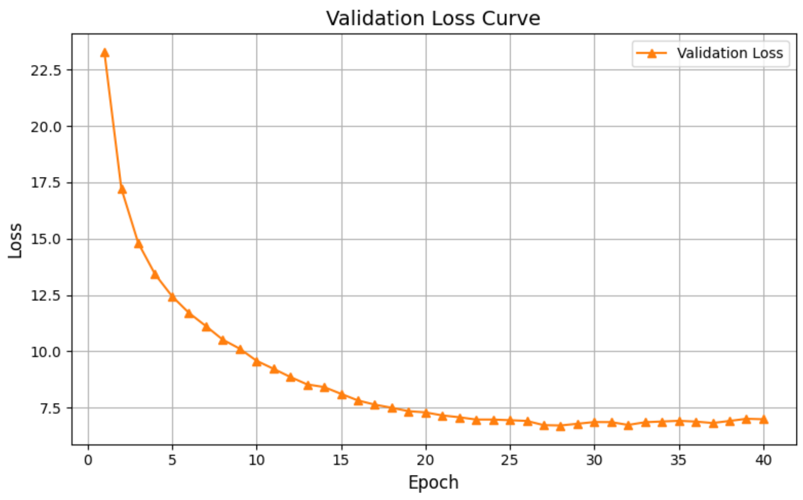
3. Experiment

In this section, I present the training process, hyperparameter tuning, and evaluation results of my SecBERT-BiLSTM-CRF model on the DNTRI dataset.

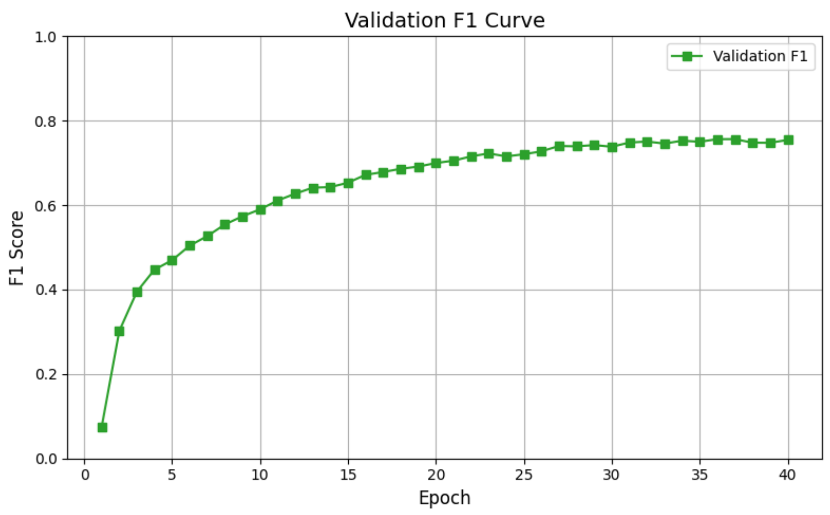
- Epochs: I set the maximum number of training epochs to 50. However, using early stopping (patience = 3), the training was stopped after 40 epochs to avoid overfitting.
- Learning Rate: I experimented with different learning rates including 3e-4, 4e-4, and 5e-5. The best performance was achieved with 5e-5, which I used in the final training.
- Scheduler: I enabled a learning rate scheduler (ReduceLROnPlateau) which monitors the validation F1 score. If the score does not improve after 2 epochs, the learning rate is reduced by a factor of 0.5. I found that using this scheduler significantly improved model performance compared to training with a constant learning rate.
- Dropout: I also tested the effect of adding a dropout layer (p=0.1) after the BiLSTM. Surprisingly, the model without dropout achieved better F1 performance, so I removed it in the final setup. (With dropout=0.1, the best F1 was 0.8127, while without dropout the F1 reached 0.8196.)
- Training Dynamics: The following figures show the trend of training loss, validation loss, and validation F1 score over 40 epochs:



- The figure shows that the training loss decreases steadily across epochs, indicating that the model is successfully learning the training patterns.
- The sharp drop at the beginning suggests that the model quickly grasps the basic structure of the data.



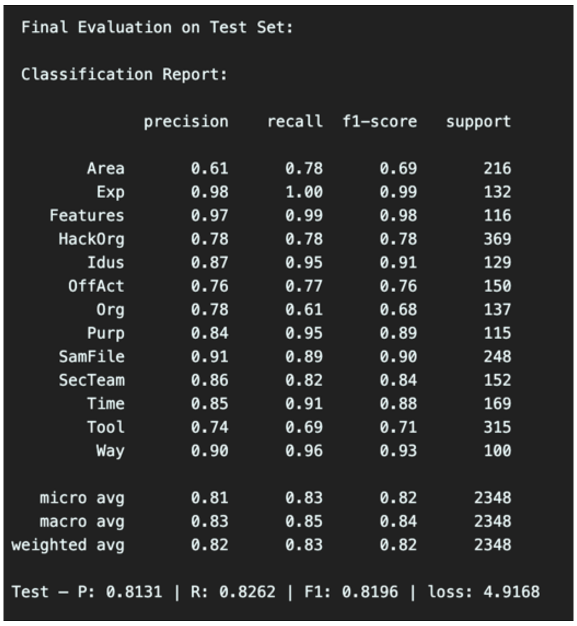
- The validation loss decreases consistently until around epoch 28 and then stabilizes, showing that the model generalizes well to unseen validation data.
- This confirms the effectiveness of early stopping, which helps prevent overfitting.



- The validation F1 score increases continuously and reaches a plateau around epoch 35, indicating strong improvement in model prediction quality.
- This curve helps determine the optimal stopping point, which aligns with the performance trends observed in loss curves.

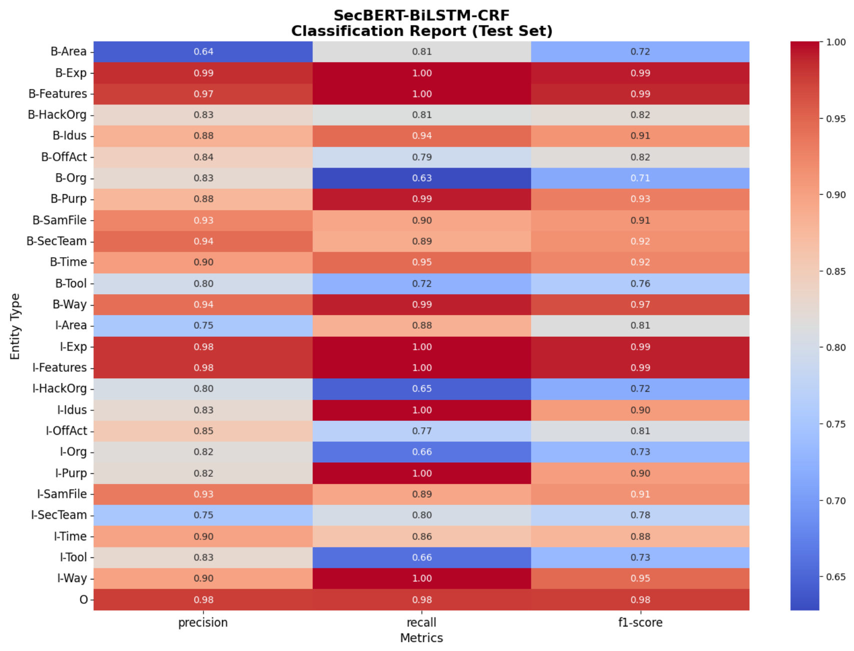
Test Results:

- Precision: 0.8131
- Recall: 0.8262
- F1-score: 0.8196



```
Final Evaluation on Test Set:

Classification Report:

              precision    recall  f1-score   support

        Area       0.61      0.78      0.69       216
         Exp       0.98      1.00      0.99       132
    Features       0.97      0.99      0.98       116
     HackOrg       0.78      0.78      0.78       369
        Idus       0.87      0.95      0.91       129
      OffAct       0.76      0.77      0.76       150
         Org       0.78      0.61      0.68       137
        Purp       0.84      0.95      0.89       115
     SamFile       0.91      0.89      0.90       248
     SecTeam       0.86      0.82      0.84       152
        Time       0.85      0.91      0.88       169
        Tool       0.74      0.69      0.71       315
         Way       0.90      0.96      0.93       100

   micro avg       0.81      0.83      0.82      2348
   macro avg       0.83      0.85      0.84      2348
weighted avg       0.82      0.83      0.82      2348

Test — P: 0.8131 | R: 0.8262 | F1: 0.8196 | loss: 4.9168
```

Entity-wise Performance:



Entity Type	precision	recall	f1-score
B-Area	0.64	0.81	0.72
B-Exp	0.99	1.00	0.99
B-Features	0.97	1.00	0.99
B-HackOrg	0.83	0.81	0.82
B-Idus	0.88	0.94	0.91
B-OffAct	0.84	0.79	0.82
B-Org	0.83	0.63	0.71
B-Purp	0.88	0.99	0.93
B-SamFile	0.93	0.90	0.91
B-SecTeam	0.94	0.89	0.92
B-Time	0.90	0.95	0.92
B-Tool	0.80	0.72	0.76
B-Way	0.94	0.99	0.97
I-Area	0.75	0.88	0.81
I-Exp	0.98	1.00	0.99
I-Features	0.98	1.00	0.99
I-HackOrg	0.80	0.65	0.72
I-Idus	0.83	1.00	0.90
I-OffAct	0.85	0.77	0.81
I-Org	0.82	0.66	0.73
I-Purp	0.82	1.00	0.90
I-SamFile	0.93	0.89	0.91
I-SecTeam	0.75	0.80	0.78
I-Time	0.90	0.86	0.88
I-Tool	0.83	0.66	0.73
I-Way	0.90	1.00	0.95
O	0.98	0.98	0.98

As shown in the heatmap, the model performs best on frequent and semantically distinct entities such as Exp, Features, and Time. However, it struggles with less frequent or overlapping classes like B-Org or I-Tool, which suggests potential label ambiguity or insufficient training data for those classes.

These results demonstrate the effectiveness of the model architecture and training strategy, though further fine-tuning could potentially improve performance on lower-scoring entity types.

4. Conclusion

In this homework, I successfully implemented a robust NER model by combining SecBERT with BiLSTM and CRF. The model benefits from domain-specific embeddings and sequential decoding, achieving an F1 score over 0.81 on the DNTRI test set. Although the performance is strong overall, further improvements may be possible via hyperparameter tuning, data augmentation, or incorporating additional regularization techniques such as dropout or adversarial training.