NLP202 - Assignment 2: Sequence Labelling

Ishika Kulkarni
ikulkar1@ucsc.edu

Abstract

This report addresses implementing a Named Entity Recognition (NER) system using a BiLSTM-CRF model designed to recognize entities from input sentences. The model incorporates both word-level and character-level information to improve performance. The dataset follows the BIO format, with sentences labeled for entity recognition. Our approach leverages PyTorch for the deep learning framework and utilizes various evaluation metrics, such as precision, recall, and F1 score.

1 Introduction

Named Entity Recognition (NER) is an essential task in Natural Language Processing (NLP) that focuses on recognizing entities such as names, locations, and organizations within text. This task is crucial for applications like information extraction and question answering. In this project, we aim to build a BiLSTM-CRF model to carry out NER on a dataset using the BIO (Begin, Inside, Outside) tagging scheme, which is widely adopted in NLP for such purposes.

The BiLSTM (Bidirectional Long Short-Term Memory) layer is designed to capture context from both past and future tokens, making it well-suited for sequence labeling tasks.

We also incorporate a CRF (Conditional Random Field) layer to enhance sequence predictions, ensuring that our predictions for each token align with the overall structure of the sentence.

2 Dataset

The dataset used for this assignment is a standard NER dataset with sentences annotated in the BIO format, where each token is labeled as belonging to a particular named entity or as a non-entity. The dataset is divided into training, validation, and test sets.

Token	Label
IL-2	B-DNA
gene	I-DNA
expression	O
NF-kappa	B-protein
B	I-protein
activation	O
through	O
CD28	B-protein
requires	O
reactive	O

Table 1: Dataset

Each sentence is a sequence of words, with corresponding labels indicating the entity type for each word. The dataset includes entities such as persons, organizations, locations, and miscellaneous entities.

The data is tokenized, and special tokens such as ¡PAD¿ are introduced for padding sentences to a uniform length. This ensures that all sentences in the batch have the same length during training.

The data is read from a file, and the words are mapped to indices using a word-to-index dictionary, while the labels are similarly mapped using a label-to-index dictionary. The training data is fed into a DataLoader to enable efficient batching and padding during model training.

3 Implementation

- **BiLSTM-CRF Model:**
  - The model uses a Bi-directional LSTM (BiLSTM) layer to capture contextual information from both the left and right of each token.
  - CRF (Conditional Random Field) is applied to the BiLSTM output for struc-

tured prediction, considering tag transitions for improved accuracy.
  - The BiLSTM is trained on the tokenized input data, capturing dependencies and providing robust feature extraction.
  - The CRF layer optimizes tag sequences by ensuring valid transitions based on training data.

- **Viterbi Algorithm:**
  - The Viterbi algorithm is used during decoding to find the most likely sequence of labels.
  - It efficiently computes the optimal sequence of tags based on transition probabilities learned during training.
  - Viterbi ensures that the output tag sequence is consistent with the model's learned parameters and constraints.

- **CNN Layer:**
  - The CNN performs convolution operations on the embedded word representations to extract local dependencies.
  - Max pooling is used to extract the most important features from the convolutional layers.
  - The output of the CNN layer is passed to the BiLSTM for sequence modeling.

- **Data Preprocessing and DataLoader:**
  - The dataset is tokenized, splitting sentences into individual tokens, and labels are assigned to each token.
  - Padding is applied to sentences to ensure uniform length for input into the model.
  - A custom Dataset class is implemented to convert tokens and labels into numerical indices using predefined vocabularies.
  - A DataLoader is used to batch and shuffle the data efficiently during training.

- **Training and Evaluation:**
  - Training is done using the Adam optimizer to minimize the negative log-likelihood loss.
  - The model's performance is evaluated using precision, recall, and F1 score on the validation set after each epoch.
  - Early stopping is used to avoid overfitting, where the model stops training if performance on the validation set stops improving.
  - The trained model is tested on unseen data (test set) to assess its generalization performance.

Hyperparameter	**Value**
Embedding Dimension	100
Hidden Dimension	256
Batch Size	2
Learning Rate	0.001
Optimizer	Adam
Epochs	5
Device	cuda (if available)

Table 2: BiLSTM-CRF Implementation Hyperparameters

Hyperparameter	**Value**
Word Embedding Dimension	100
Character Embedding Dimension	30
Character Kernel Size	3
Character Filters (Hidden Dim)	50
Hidden Dimension (LSTM)	256
Learning Rate	0.001
Optimizer	Adam

Table 3: Character Level CNN Hyperparameters

Hyperparameter	**Value**
Word Embedding Dimension	100
Hidden Dimension (LSTM)	256
LSTM Layers	1 (Bidirectional)
Batch Size	16
Learning Rate	0.001
Optimizer	Adam
Epochs	20
Device	cuda (if available)

Table 4: BiLSTM-CRF with Multiple Loss Functions Hyperparameters

4 Experiment Outputs

- **Implementation 1: BiLSTM-CRF**
  - **Dev Output:**
    - from - O
    - activated - B-cell_type
    - peripheral - I-cell_type
    - blood - I-cell_type
    - mononuclear - I-cell_type
  - **Test Output:**
    - glucocorticoid - B-protein
    - receptors - I-protein
    - in - O
    - lymphocytes - O
    - and - O
- **Implementation 2: Character Level CNN**
  - **Dev Output:**
    - apoptosis - B-DNA
    - , - O
    - confirming - O
    - the - O
    - glucocorticoid - B-protein
  - **Test Output:**
    - cells - O
    - were - O
    - predominantly - O
    - IL-2 - I-cell_line
    - , - O
- **Implementation 3: BiLSTM CRF and CNN**
  - **Dev Output:**
    - in - O
    - peripheral - B-cell_type
    - blood - I-cell_type
    - mononuclear - I-cell_type
    - cells - I-cell_type
  - **Test Output:**
    - and - O
    - T - B-cell_type
    - cells - I-cell_type
    - from - O
    - IkappaBalpha - B-protein

5 Results

Model	Dev F1	Test F1
BiLSTM - CRF	0.8539	0.8201
Character Level CNN	0.4896	0.4702
BiLSTM CRF and CNN	0.89	0.87

Table 5: Model Performance Results (F1 Score)

6 Conclusion

- **Performance Comparison:** The BiLSTM-CRF model performed the best, achieving the highest F1 scores on both dev and test sets (dev F1 = 0.8539, test F1 = 0.8201), showcasing its ability to effectively capture sequential dependencies and provide accurate predictions for named entity recognition tasks.
- **Character-Level CNN:** The character-level CNN model exhibited lower performance, with a dev F1 of 0.4896 and a test F1 of 0.4702. This suggests that while the model captures word-level features through CNNs, it struggles to fully grasp the complexities of sequential and contextual relationships in the data.
- **Combined BiLSTM-CRF and CNN Model:** The hybrid BiLSTM-CRF and CNN model outperformed the individual models, with the highest dev and test F1 scores (dev F1 = 0.89, test F1 = 0.87). This result highlights the effectiveness of combining both character-level CNN and BiLSTM-CRF components to leverage the strengths of both word-level and sequence-level information.