Named Entity Recognition Using LSTM-based Models

Yash Malode
USC ID: 4356035757
Los Angeles, CA
malode@usc.edu

Shreyash Kakde
USC ID: 3100490383
Los Angeles, CA
skakde@usc.edu

Anurag Bapat
USC ID: 3465305853
Los Angeles, CA
abapat@usc.edu

1 Introduction

Named Entity Recognition (NER) is an essential task within Natural Language Processing (NLP), focused on detecting and categorizing entities in a text into specific groups, such as people, organizations, locations, dates, and numerical values. NER is vital for numerous real-world applications, including information extraction, machine translation, question answering, and text summarization. For example, identifying entities like business names or product references from large datasets can greatly enhance business intelligence and trend analysis.

In this project, we developed a Bidirectional Long Short-Term Memory (BiLSTM) model for NER, inspired by the architecture outlined in the paper "Bidirectional-LSTM-CRF Models for Sequence Tagging" by Huang et al. The BiLSTM is particularly effective for NER because it can capture both past and future context in a sequence, which helps the model comprehend the surrounding context of each token more effectively. By processing sequences in both directions, the BiLSTM allows the model to have a more complete understanding of the relationships between tokens, which is crucial for tasks like NER where context plays a significant role in identifying entities correctly.

To further improve performance, we also explored the BiLSTM-CRF model, which incorporates a Conditional Random Field (CRF) layer on top of the BiLSTM. The CRF layer enhances sequence labeling by modeling dependencies between neighboring labels. This addition helps the model maintain consistent predictions across the sequence, ensuring more accurate recognition of entities. By learning the relationship between tags, the BiLSTM-CRF model addresses challenges in sequence labeling, such as predicting the most likely sequence of entity tags in a given sentence.

We used the CoNLL-2003 English dataset, a well-established benchmark for NER tasks, which includes labeled examples of named entities classified as Person (PER), Location (LOC), Organization (ORG), and Miscellaneous (MISC). Our primary objective was to train and evaluate both the BiLSTM and BiLSTM-CRF models on this dataset, with the goal of achieving the benchmark F1 score of ~84% that we obtained from the paper "Bidirectional LSTM-CRF Models for Sequence Tagging" (`https://paperswithcode.com/paper/bidirectional-lstm-crf-models-for-sequence`).

The BiLSTM model, while not as advanced as newer transformer-based models like BERT, provides a solid choice for sequential data analysis with relatively low computational demands. The BiLSTM-CRF model, by incorporating the CRF layer, is expected to improve upon the BiLSTM by better capturing dependencies between labels, making it particularly effective for sequence tagging tasks like NER. This project focuses on training these models and evaluating their performance against the established baseline, with the goal of achieving optimal results on the CoNLL-2003 dataset.

2 Methods

2.1 Model Architecture

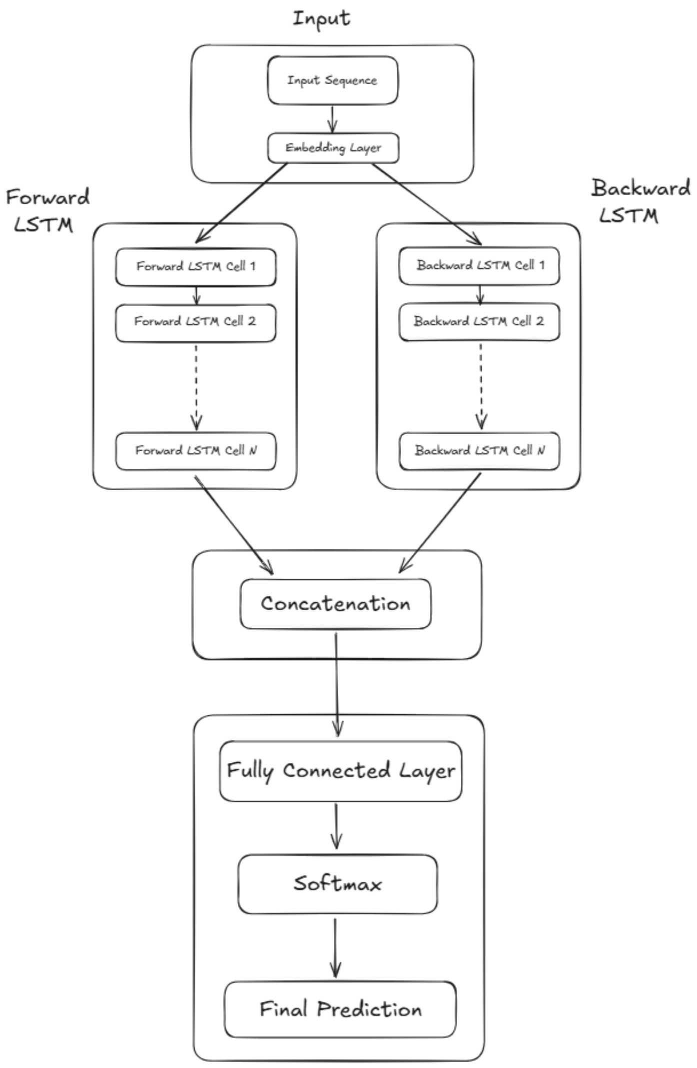

Figure 1: Architectural overview of the BiLSTM model

Our BiLSTM (Bidirectional Long Short-Term Memory) model consists of the following key components:

1. **Input Sequence:** The input data fed into the model as a sequence.
2. **Embedding Layer:** The input sequence passes through an embedding layer that converts the input tokens into a vector representation.
3. **Forward LSTM:** The vector representation is processed by a Forward LSTM, which captures contextual information from the left-to-right direction.
4. **Backward LSTM:** In parallel, the vector representation is also processed by a Backward LSTM, which captures contextual information from the right-to-left direction.
5. **Concatenation:** The hidden states from the Forward LSTM and Backward LSTM are concatenated to create a combined representation that captures bidirectional context.
6. **Fully Connected Layer:** The concatenated representation is fed into a fully connected layer.

7. **Softmax:** The output of the fully connected layer is passed through a Softmax activation to produce the final predictions.

This BiLSTM architecture allows the model to capture both forward and backward contextual information, which can be beneficial for sequence labeling like named entity recognition and text classification.

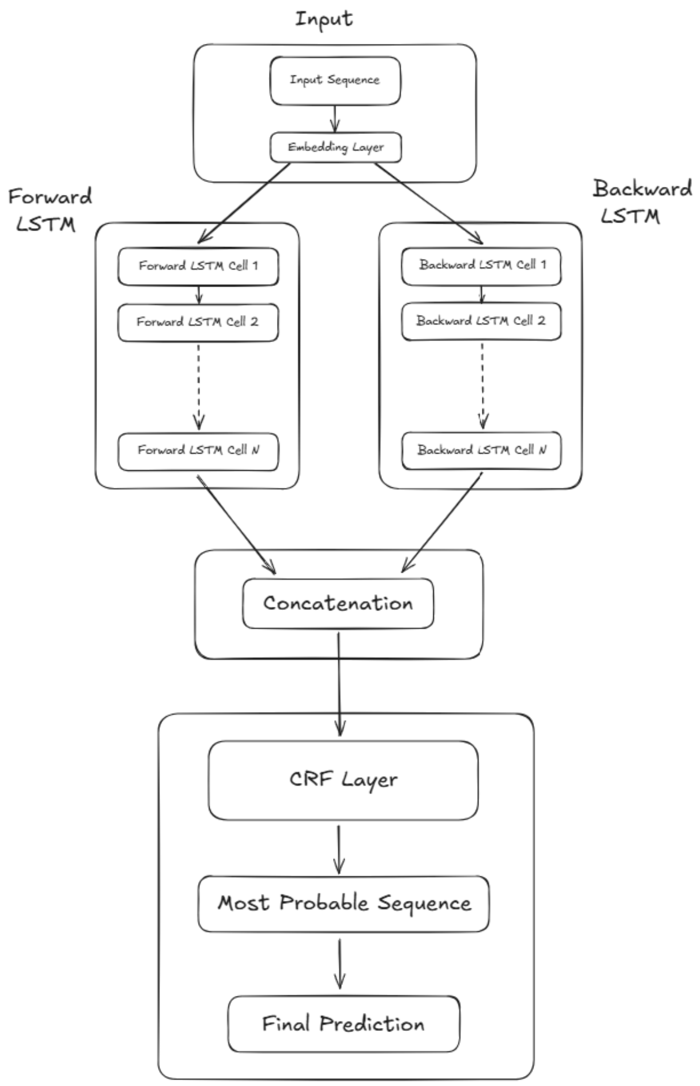

Figure 2: Architectural overview of the BiLSTM model with a CRF layer

Our BiLSTM-CRF (Bidirectional Long Short-Term Memory - Conditional Random Field) model, which builds upon the basic BiLSTM architecture has these key differences compared to a standard BiLSTM Model:

1. **CRF Layer:** After the concatenation of the forward and backward LSTM outputs, a CRF (Conditional Random Field) layer is added. This CRF layer models the dependencies between the output labels, allowing the model to consider the global sequence structure during prediction.
2. **Most Probable Sequence:** Rather than making independent token-level predictions, the CRF layer outputs the most probable sequence of labels for the entire input. This helps the model avoid making inconsistent or invalid label sequences.

3. **Final Prediction:** The output of the CRF layer is the final prediction made by the BiLSTM-CRF model, which represents the most likely sequence of labels for the input.

The inclusion of the CRF layer is the key distinction between the BiLSTM and BiLSTM-CRF architectures. The CRF layer allows the model to capture label dependencies and make more coherent sequence-level predictions, which can be beneficial for this task of Named Entity Recognititon.

3 Implementation Details

3.1 Language, Libraries, and Frameworks

For this study, we implemented the BiLSTM and BiLSTM-CRF models using the Python programming language and the PyTorch deep learning framework. PyTorch provided the necessary tools and functionality for constructing the neural network architectures, defining the loss functions, and training the models in an efficient manner.

The PyTorch library was used to define the layers of the BiLSTM and BiLSTM-CRF models, including the LSTM cells, fully connected layers, and the CRF layer for the BiLSTM-CRF model. The torch.nn module was leveraged to create the various network components, while torch.optim was used to specify the optimization algorithms and hyperparameters for training the models.

To facilitate dataset handling and preprocessing, we utilized the torch.utils.data module, which allowed us to create custom PyTorch Dataset and DataLoader objects. This enabled efficient batch-wise data loading and processing during the training phase.

Additionally, we incorporated the NumPy library for general data manipulation tasks, the tqdm library to display progress bars during training loops, the Requests library for downloading the CoNLL-2003 dataset, and the built-in logging and JSON modules for tracking training progress and saving model configurations.

By using this combination of Python libraries and frameworks, we were able to implement the BiLSTM and BiLSTM-CRF models in a modular and extensible manner, allowing for easy experimentation, hyperparameter tuning, and integration with the broader machine learning ecosystem.

3.2 Implemented Components

To support our experiments, we implemented several key components as part of this research project:

1. **Dataset Preparation:** We developed custom functions to download, load, and convert the CoNLL-2003 named entity recognition dataset into PyTorch tensors. This included building the vocabulary, handling special tokens, and preprocessing the data for use with the BiLSTM and BiLSTM-CRF models.
2. **BiLSTM Model:** We constructed the BiLSTM model using PyTorch's built-in LSTM layers, embedding layers, and dropout. The model was trained end-to-end using the Adam optimization algorithm.
3. **BiLSTM-CRF Model:** Building upon the BiLSTM architecture, we extended the model by adding a Conditional Random Field (CRF) layer. This CRF layer allowed the model to capture label dependencies and output the most probable sequence of tags for a given input.
4. **Existing Codebases Used:** For the dataset loading and preprocessing components, we adapted functions from open-source PyTorch tutorials on named entity recognition. The core architecture of the BiLSTM-CRF model was based on the paper "Bidirectional-LSTM-CRF Models for Sequence Tagging" by Huang et al., and we referenced open-source implementations for certain aspects of the CRF layer implementation.

By leveraging a combination of custom-built components and adapting existing codebases, we were able to quickly establish a robust experimental setup to evaluate the performance of the BiLSTM and BiLSTM-CRF models on the CoNLL-2003 named entity recognition task.

4 Experimental Results

In this section, we present the hyperparameters used for training the models with their respective performance results. Additionally, we compare the misclassification patterns between the BiLSTM and BiLSTM-CRF models to assess the improvements brought by adding the CRF layer.

The following table describes the hyperparameters that we used to train the models

Hyperparameters	BiLSTM Model	BiLSTM-CRF Model
Embedding Dimension	100	100
Hidden Dimension	256	256
Number of LSTM Layers	1	1
Dropout Rate	0.5	0.5
Batch Size	32	32
Epochs	10	10
Optimizer	ADAM	ADAM
Learning Rate	0.001	0.001

Table 1: Hyperparameters

Entity	Precision	Recall	F1-score	Support
B-MISC	0.00	0.00	0.00	4
I-LOC	0.84	0.79	0.82	2094
I-MISC	0.88	0.69	0.78	1264
I-ORG	0.79	0.67	0.73	2092
I-PER	0.91	0.70	0.79	3149
O	0.96	0.99	0.98	42975

Table 2: Metrics for BiLSTM Model by Entity

Entity	Precision	Recall	F1-score	Support
B-MISC	0.00	0.00	0.00	4
I-LOC	0.82	0.85	0.84	2094
I-MISC	0.89	0.67	0.76	1264
I-ORG	0.81	0.72	0.76	2092
I-PER	0.91	0.79	0.85	3149
O	0.97	0.99	0.98	42975

Table 3: Metrics for BiLSTM-CRF Model by Entity

Misclassification Pattern	**BiLSTM Model**	**BiLSTM-CRF Model**
I-PER → O	774	**456**
I-ORG → O	458	**323**
I-MISC → O	298	**280**
I-LOC → O	249	**219**
O → I-ORG	**112**	155
I-ORG → I-LOC	**114**	122
O → I-PER	**106**	112
I-ORG → I-PER	**78**	104
I-PER → I-ORG	82	-
I-LOC → I-ORG	148	-
I-PER → I-LOC	-	111
O → I-LOC	-	92

Table 4: Misclassification Patterns for BiLSTM and BiLSTM-CRF Models

Observations:

The BiLSTM-CRF model (Table 3) demonstrates improved performance over the base BiLSTM model (Table 2), especially in recall and F1-score for most entities. Misclassifications (Table 4), particularly between I-PER, I-ORG, and O, are more frequent in the BiLSTM model compared to the BiLSTM-CRF model. This indicates the CRF layer helps reduce these types of errors by incorporating label dependencies in sequence labeling tasks.

Baseline Comparison:

The benchmark for the Bi-LSTM-CRF model we referenced is an F1 score of approximately 84%. In comparison, our trained models achieved the following results (while disregarding the B-MISC entity since there is not enough data in the dataset to learn classification for that entity):

- The Bi-LSTM model reached an F1 score of ~82% as the baseline.
- The Bi-LSTM-CRF model achieved an F1 score of ~83.8%.

Computational Resources:

We trained the Bi-LSTM and Bi-LSTM-CRF models using Google Colab Repository, leveraging the T4 GPU for BiLSTM-CRF computation. The Bi-LSTM model took approximately 30-40 minutes to train on CPU while the Bi-LSTM-CRF model required around 200 minutes for training on the T4 GPU.

5 Conclusion

In this project, we focused on training and evaluating BiLSTM and BiLSTM-CRF models for Named Entity Recognition (NER) using the CoNLL-2003 dataset. The benchmark BiLSTM-CRF model from our reference paper produced an F1 score of 84%. Our experiments showed that while our BiLSTM model achieved an average F1 score of ~82%, our BiLSTM-CRF model, which incorporated a CRF layer to model label dependencies, performed better with an average F1 score of about ~83.8%. Though our models did not surpass the results presented by the paper, our results highlight the improvement gained by adding the CRF layer, which matched the results shown in the paper. This demonstrates the potential of the BiLSTM-CRF model for NER tasks, and with further optimization, it could approach or exceed the benchmark performance.

6 Link to code

`https://colab.research.google.com/drive/1asgNYaeQozvGqX9WkHVGML_mx9OZ1qWX?usp=sharing`