Named Entity Recognition (NER) Model with Machine Learning

Naveenreddy Narayana Jaya Prakash Narayana Raavi Sai Narendra Reddy L
University of Memphis, Memphis, Tennessee, USA
{nnrayana, jraavi, slkkrddy}@memphis.edu

Abstract

Many uses of natural language processing (NLP) require the ability to precisely identify and categorize named entities in text data. A fundamental NLP task called Named Entity Recognition (NER) involves extracting entities from text, like names of people, places, organizations, and temporal expressions, and classifying them into predefined categories. This paper provides a thorough introduction to neural network analysis (NER), covering its importance, methods, and applications in a range of fields. We discuss state-of-the-art methods for machine learning as well as new developments powered by deep learning technologies, especially those that leverage contextual embeddings from transformer-based models such as BERT. We also look at the difficulties encountered in NER, such as entity disambiguation and managing language variations specific to a given domain. We also touch on the useful applications of NER for improving information extraction, document processing automation, and semantic understanding. We hope to demonstrate through this investigation how important NER is to the development of NLP systems and how it can revolutionize data-driven insights in a variety of industries.

1 Introduction

Named Entity Recognition (NER) is a crucial component of Natural Language Processing (NLP) that involves identifying and classifying named entities in text into predefined categories such as names of persons, organizations, locations, expressions of times, quantities, monetary values, percentages, etc. This technology serves as the backbone for a variety of applications, including information retrieval, knowledge extraction, question answering, and machine translation, among others. The NER model operates by analyzing the context in which a word appears within a sentence or across a text corpus to determine its identity as a named entity. Advanced techniques involving machine learning, particularly deep learning models like LSTM (Long Short-Term Memory) networks or more recent transformer-based models such as BERT (Bidirectional Encoder Representations from Transformers), have significantly improved the accuracy and efficiency of named entity recognition systems.

• Issue: The project focuses on Named Entity Recognition (NER), a crucial task in Natural Language Processing (NLP) that involves identifying real-world objects such as persons, places, organizations, and products in each text.

• Importance: NER is essential for various applications, including information extraction, question answering, and sentiment analysis. It enhances the understanding of text by identifying and categorizing named entities.

• If we want to read a lot of news articles and find important names, places, companies, and other special details, then this helps in making summaries, finding articles easily, and understanding people's feelings about different topics.

2 Related Work

Neural Architectures for Nested NER through Linearization:
Paper Link: https://aclanthology.org/P19-1527.pdf

In this paper, the authors propose two different neural network architectures for nested named entity recognition (NER), a setting where named entities can overlap and be labeled with more than one label.

The first architecture encodes the nested labels using a linearized scheme. In the second architecture, the nested NER is viewed as a sequence-to-sequence (seq2seq) task. The authors evaluate their models on four corpora: ACE-2004, ACE-2005, GENIA, and Czech CNEC. They also compare their models to other state-of-the-art methods. The proposed seq2seq architecture outperforms all other models on the nested NER task. The authors achieve further improvements by adding contextual word embeddings to their models.

Their work is relevant to my research because it investigates novel neural architectures for nested NER. Their seq2seq architecture achieves state-of-the-art performance on this task, suggesting that this approach is well-suited for nested NER. Additionally, their use of contextual word embeddings further improves the performance of their models. This suggests that incorporating contextual information can be beneficial for nested NER tasks.

We plan to build upon their work by exploring different seq2seq architectures for nested NER. We will also investigate the use of different contextual word embedding techniques. Additionally, We will explore the application of these techniques to other NLP tasks that involve nested structures, such as semantic role labeling and parsing.

3 Data

3.1 Dataset Description

Data Source: The dataset used for training and testing the NER model is sourced from Kaggle. (https://www.kaggle.com/datasets/rohitr4307/ner-dataset)

Preprocessing: The data will be loaded using pandas, and necessary modifications will be made to prepare it for training a Neural Network. This includes mapping tokens and tags, transforming columns, and splitting the data into training, validation, and test sets.

3.2 Overview of the Dataset

• Sentence ID: An identifier for each sentence in the dataset.
• Word: The individual words present in the sentences.
• POS (Part of Speech): The grammatical category of each word (e.g., noun, verb, adjective).
• Tag: The corresponding NER tag assigned to each word (e.g., person, organization, location).
• The Data Frame comprises 47,959 entries with no missing values across the four columns.
• Each column represents crucial aspects of the dataset necessary for NER, including identifying words, their corresponding parts of speech, and named entity tags.

3.3 Data Statistics

The dataset contains a diverse range of words, parts of speech, and named entity tags.There are 47,959 unique Sentence IDs, indicating one unique sentence per entry.
The 'Word' column has 47,575 unique words, with 'Sentence: 1' being the most frequent entry.Similarly, the 'POS' column has 47,214 unique part-of-speech tags, with 'NNP' (proper noun, singular) being the most frequent.
The 'Tag' column has 33,318 unique named entity tags, with 'O' (denoting 'Other') being the most frequent.
The total number of classes are 17
I-org I-art I-geo I-gpe I-nat B-org I-tim B-nat B-tim B-eve B-per I-per I-eve B-geo B-art O B-gpe

3.4 Data Preprocessing

The script begins by loading a CSV file that contains the NER dataset. Each entry in the dataset is expected to include words, their corresponding Part-of-Speech (POS) tags, and the NER tags. The dataset is pre-processed by converting string representations of lists into actual Python lists using the ast.literal eval function.
Unique words and tags are then extracted to form a vocabulary (word to ix) and a set of labels (tag to ix), respectively, with special tokens added for padding and unknown words. These mappings are critical for converting textual data into numerical form that can be processed by the neural network.

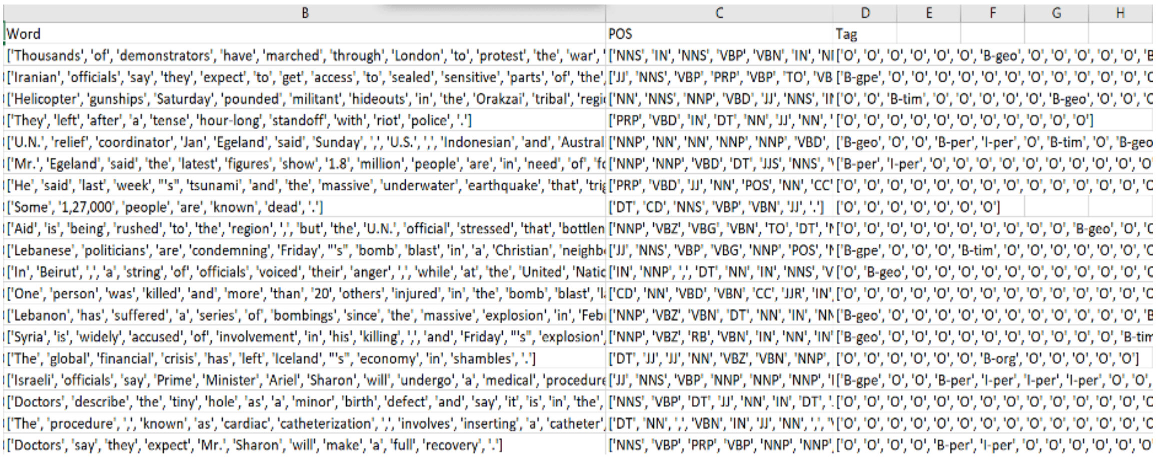
3.5 Dataset and DataLoader

A custom NERDataset class is defined to handle the input data for the PyTorch model. It takes sentences and tags, converts them to indices using the previously defined mappings, and wraps them in a Dataset object that can be used by a DataLoader. The DataLoader utilizes the pad collate function to dynamically pad the sentences and tags to the same length within each batch, which is necessary for efficient batch processing.

4 Methods

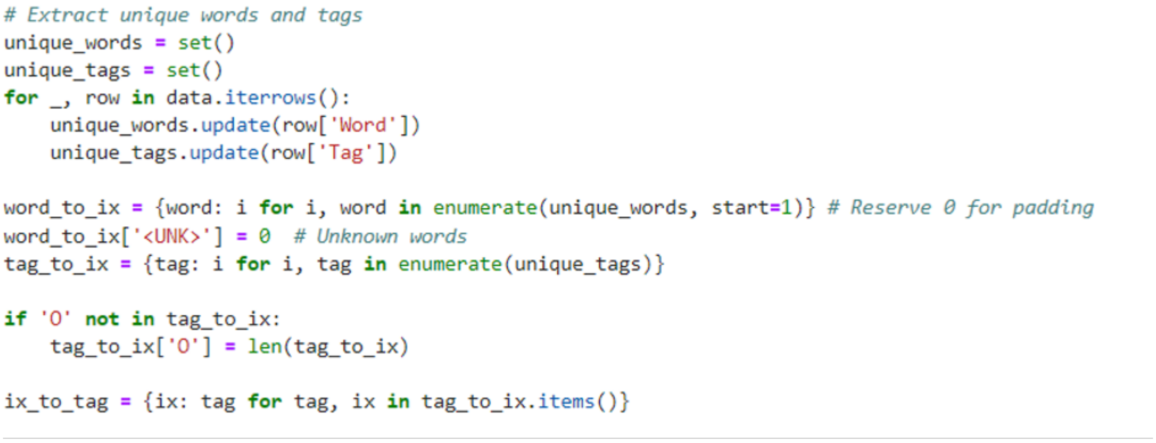
4.1 Methodology

Named Entity Recognition (NER) can be approached using a variety of architectures and techniques, each with special benefits and working



B	C	D	E	F	G	H
Word	POS	Tag				
['Thousands', 'of', 'demonstrators', 'have', 'marched', 'through', 'London', 'to', 'protest', 'the', 'war', '	['NNS', 'IN', 'NNS', 'VBP', 'VBN', 'IN', 'NI	['O', 'O', 'O', 'O', 'O', 'O', 'B-geo', 'O', 'O', 'O', 'O', 'O', 'B				
['Iranian', 'officials', 'say', 'they', 'expect', 'to', 'get', 'access', 'to', 'sealed', 'sensitive', 'parts', 'of', 'the',	['JJ', 'NNS', 'VBP', 'PRP', 'VBP', 'TO', 'VB	['B-gpe', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'C				
['Helicopter', 'gunships', 'Saturday', 'pounded', 'militant', 'hideouts', 'in', 'the', 'Orakzai', 'tribal', 'regi	['NN', 'NNS', 'NNP', 'VBD', 'JJ', 'NNS', 'IN	['O', 'O', 'B-tim', 'O', 'O', 'O', 'O', 'O', 'B-geo', 'O', 'O', 'C				
['They', 'left', 'after', 'a', 'tense', 'hour-long', 'standoff', 'with', 'riot', 'police', '.']	['PRP', 'VBD', 'IN', 'DT', 'NN', 'JJ', 'NN', '	['O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O']				
['U.N.', 'relief', 'coordinator', 'Jan', 'Egeland', 'said', 'Sunday', ',', 'U.S.', ',', 'Indonesian', 'and', 'Austral	['NNP', 'NN', 'NN', 'NNP', 'NNP', 'VBD',	['B-geo', 'O', 'O', 'B-per', 'I-per', 'O', 'B-tim', 'O', 'B-geo				
['Mr.', 'Egeland', 'said', 'the', 'latest', 'figures', 'show', '1.8', 'million', 'people', 'are', 'in', 'need', 'of', 'fc	['NNP', 'NNP', 'VBD', 'DT', 'JJS', 'NNS', '	['B-per', 'I-per', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O				
['He', 'said', 'last', 'week', "'s", 'tsunami', 'and', 'the', 'massive', 'underwater', 'earthquake', 'that', 'trig	['PRP', 'VBD', 'JJ', 'NN', 'POS', 'NN', 'CC'	['O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'C				
['Some', '1,27,000', 'people', 'are', 'known', 'dead', '.']	['DT', 'CD', 'NNS', 'VBP', 'VBN', 'JJ', '.']	['O', 'O', 'O', 'O', 'O', 'O', 'O']				
['Aid', 'is', 'being', 'rushed', 'to', 'the', 'region', ',', 'but', 'the', 'U.N.', 'official', 'stressed', 'that', 'bottlen	['NNP', 'VBZ', 'VBG', 'VBN', 'TO', 'DT', 'N	['O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'B-geo', 'O', 'C				
['Lebanese', 'politicians', 'are', 'condemning', 'Friday', "'s", 'bomb', 'blast', 'in', 'a', 'Christian', 'neighb	['JJ', 'NNS', 'VBP', 'VBG', 'NNP', 'POS', 'N	['B-gpe', 'O', 'O', 'O', 'B-tim', 'O', 'O', 'O', 'O', 'O', 'O', 'C				
['In', 'Beirut', ',', 'a', 'string', 'of', 'officials', 'voiced', 'their', 'anger', ',', 'while', 'at', 'the', 'United', 'Natic	['IN', 'NNP', ',', 'DT', 'NN', 'IN', 'NNS', 'V	['O', 'B-geo', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'C				
['One', 'person', 'was', 'killed', 'and', 'more', 'than', '20', 'others', 'injured', 'in', 'the', 'bomb', 'blast', 'l	['CD', 'NN', 'VBD', 'VBN', 'CC', 'JJR', 'IN',	['O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'C				
['Lebanon', 'has', 'suffered', 'a', 'series', 'of', 'bombings', 'since', 'the', 'massive', 'explosion', 'in', 'Feb	['NNP', 'VBZ', 'VBN', 'DT', 'NN', 'IN', 'NN	['B-geo', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'B				
['Syria', 'is', 'widely', 'accused', 'of', 'involvement', 'in', 'his', 'killing', ',', 'and', 'Friday', "'s", 'explosion',	['NNP', 'VBZ', 'RB', 'VBN', 'IN', 'NN', 'IN'	['B-geo', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'B-tim				
['The', 'global', 'financial', 'crisis', 'has', 'left', 'Iceland', "'s", 'economy', 'in', 'shambles', '.']	['DT', 'JJ', 'JJ', 'NN', 'VBZ', 'VBN', 'NNP',	['O', 'O', 'O', 'O', 'O', 'O', 'B-org', 'O', 'O', 'O', 'O', 'O']				
['Israeli', 'officials', 'say', 'Prime', 'Minister', 'Ariel', 'Sharon', 'will', 'undergo', 'a', 'medical', 'procedure	['JJ', 'NNS', 'VBP', 'NNP', 'NNP', 'NNP', 'I	['B-gpe', 'O', 'O', 'B-per', 'I-per', 'I-per', 'I-per', 'O', 'O',				
['Doctors', 'describe', 'the', 'tiny', 'hole', 'as', 'a', 'minor', 'birth', 'defect', 'and', 'say', 'it', 'is', 'in', 'the',	['NNS', 'VBP', 'DT', 'JJ', 'NN', 'IN', 'DT', '.	['O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'C				
['The', 'procedure', ',', 'known', 'as', 'cardiac', 'catheterization', ',', 'involves', 'inserting', 'a', 'catheter',	['DT', 'NN', ',', 'VBN', 'IN', 'JJ', 'NN', ',', '	['O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'C				
['Doctors', 'say', 'they', 'expect', 'Mr.', 'Sharon', 'will', 'make', 'a', 'full', 'recovery', '.']	['NNS', 'VBP', 'PRP', 'VBP', 'NNP', 'NNP',	['O', 'O', 'O', 'O', 'B-per', 'I-per', 'O', 'O', 'O', 'O', 'O', 'O				

Figure 1: Overview of the dataset.



```
# Extract unique words and tags
unique_words = set()
unique_tags = set()
for _, row in data.iterrows():
    unique_words.update(row['Word'])
    unique_tags.update(row['Tag'])

word_to_ix = {word: i for i, word in enumerate(unique_words, start=1)} # Reserve 0 for padding
word_to_ix['<UNK>'] = 0  # Unknown words
tag_to_ix = {tag: i for i, tag in enumerate(unique_tags)}

if 'O' not in tag_to_ix:
    tag_to_ix['O'] = len(tag_to_ix)

ix_to_tag = {ix: tag for tag, ix in tag_to_ix.items()}
```

Figure 2: Data Preprocessing

mechanisms. In this work, we explore four different approaches: Transformer, Bidirectional Long Short-Term Memory (BiLSTM), Convolutional Neural Network (CNN), and Transfer Learning with BERT models.

4.2 BiLSTM

The methodology for implementing Named Entity Recognition (NER) using a BiLSTM neural network follows a systematic approach encompassing data preprocessing, model design, training, and evaluation
Model Architecture: The model defined is a BiLSTMForNER, a neural network comprising an embedding layer, a Bidirectional LSTM layer, and a linear layer. The embedding layer translates the input word indices into dense vectors of a specified size. The BiLSTM layer processes the embeddings in both forward and reverse directions to capture context from both the past and the future within a sequence. The output of the BiLSTM layer is passed through a linear layer that projects the LSTM outputs to the space of the tag set size, producing a distribution over possible tags.

4.3 CNN Model

Model Architecture: The CNNForNER class defines the CNN model architecture, which includes an embedding layer to convert word indices into dense embeddings, a 1D convolutional layer to extract features from the embeddings, a ReLU activation function to introduce non-linearity, and a fully connected layer to map the extracted features to the tag space. The model uses log-softmax to output the likelihood of each tag for each word in the sentence.

4.4 Transformer Model

Model Architecture: The TransformerForNER class defines the model architecture consisting of an embedding layer, positional encoding, a transformer encoder, and a final linear layer. The embedding layer maps word indices to dense vectors. The positional encoding is added to the embedding output to provide the model with information about the order of words. The transformer encoder, composed of multiple encoder layers, processes the sequence with self-attention mechanisms. The output is then passed through a linear layer to obtain the predicted tag scores.

4.5 Bert Model

Tokenization and Dataset Preparation: The BertTokenizer is used to tokenize the words. Due to BERT's WordPiece tokenization method, words may be split into subwords. Tokens are processed sentence-wise, with special tokens [CLS] at the beginning and [SEP] at the end of each tokenized sentence. The NERDataset class encapsulates the tokenization and encoding process, also ensuring that all token sequences are padded to a uniform maximum length.
Model and Optimizer: BertForTokenClassification is adapted for the NER task with a specified number of labels equal to the number of unique tags. The AdamW optimizer is employed with a learning rate suitable for fine-tuning BERT models.

5 Experiments

5.1 BiLSTM

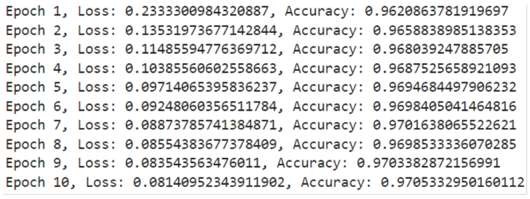
Training Procedure: The model is trained over several epochs. In each epoch, the model parameters are updated to minimize the CrossEntropyLoss between the predicted tag distributions and the actual tags. The loss is backpropagated through the network using the Adam optimizer, a popular choice for training deep neural networks due to its adaptive learning rate properties.

5.2 CNN Model

Training Procedure: The training loop involves iterating over the dataset for a fixed number of epochs. In each epoch, the model's weights are updated using backpropagation to minimize the cross-entropy loss between the predicted and true tags. The Adam optimizer is used due to its efficient computation and lower memory requirement, as well as its adaptive learning rate features which make it well-suited for training deep learning models.

5.3 Transformer model

Training Process: The training loop iterates over the dataset, with each epoch consisting of a forward pass followed by a loss calculation. Backpropagation is used to update the weights of the model with the aim of minimizing the cross-entropy loss between the predicted tags and the actual tags.



```
Epoch 1, Loss: 0.2333300984320887, Accuracy: 0.9620863781919697
Epoch 2, Loss: 0.13531973677142844, Accuracy: 0.9658838985138353
Epoch 3, Loss: 0.11485594776369712, Accuracy: 0.968039247885705
Epoch 4, Loss: 0.10385560602558663, Accuracy: 0.9687525658921093
Epoch 5, Loss: 0.09714065395836237, Accuracy: 0.9694684497906232
Epoch 6, Loss: 0.09248060356511784, Accuracy: 0.9698405041464816
Epoch 7, Loss: 0.08873785741384871, Accuracy: 0.9701638065522621
Epoch 8, Loss: 0.08554383677378409, Accuracy: 0.9698533336070285
Epoch 9, Loss: 0.083543563476011, Accuracy: 0.9703382872156991
Epoch 10, Loss: 0.08140952343911902, Accuracy: 0.9705332950160112
```

Figure 3: Epochs of BiLSTM .

Optimization: The Adam optimizer is utilized for adjusting the weights of the model, which is a popular choice in training deep learning models due to its effectiveness in handling sparse gradients and adapting the learning rate for each parameter.

5.4 Bert Model

Training Loop: The model is trained for a number of epochs, where in each epoch, the model's parameters are updated to minimize the loss function. After each batch, the optimizer's gradient is zeroed to prevent accumulation from previous iterations. The total loss is computed and printed after each epoch.

6 Results

6.1 BiLSTM

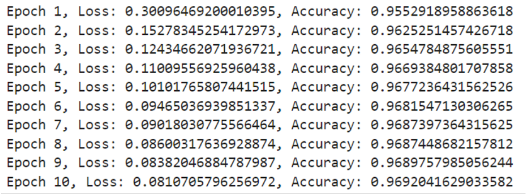
After each epoch, the model's performance is evaluated on a separate test set. The evaluate model function computes predictions for the test data without updating the model's weights (as the model is in evaluation mode). It then flattens the predictions and true tags to calculate the accuracy and produce a classification report.
The Name Entity Recognition model by using BiLSTM has been run for 10 epoch runs where we can see that the accuracy has gone up to 97.05

6.2 CNN Model

The evaluate model function assesses the model's performance by predicting tags on the test set without updating the weights. It calculates the accuracy and generates a detailed classification report.
The Name Entity Recognition model by using CNN model has been run for 10 epoch runs where we can see that the accuracy has gone up to 96.92

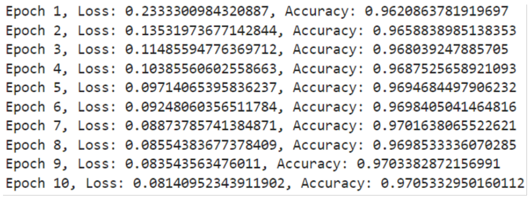
6.3 Transformer Model

The evaluate model function measures the model's performance using the test data. It calculates accuracy and generates a classification report. The Name Entity Recognition model by using Trans-



```
Epoch 1, Loss: 0.30096469200010395, Accuracy: 0.9552918958863618
Epoch 2, Loss: 0.15278345254172973, Accuracy: 0.962525145742671
Epoch 3, Loss: 0.12434662071936721, Accuracy: 0.9654784875605551
Epoch 4, Loss: 0.11009556925960438, Accuracy: 0.9669384801707858
Epoch 5, Loss: 0.10101765807441515, Accuracy: 0.9677236431562526
Epoch 6, Loss: 0.09465036939851337, Accuracy: 0.9681547130306265
Epoch 7, Loss: 0.09018030775566464, Accuracy: 0.9687397364315625
Epoch 8, Loss: 0.08600317636928874, Accuracy: 0.9687448682157812
Epoch 9, Loss: 0.08382046884787987, Accuracy: 0.9689757985056244
Epoch 10, Loss: 0.0810705796256972, Accuracy: 0.9692041629033582
```

Figure 4: Epochs of CNN Model.



```
Epoch 1, Loss: 0.2333300984320887, Accuracy: 0.9620863781919697
Epoch 2, Loss: 0.13531973677142844, Accuracy: 0.9658838985138353
Epoch 3, Loss: 0.11485594776369712, Accuracy: 0.968039247885705
Epoch 4, Loss: 0.10385560602558663, Accuracy: 0.9687525658921093
Epoch 5, Loss: 0.09714065395836237, Accuracy: 0.9694684497906232
Epoch 6, Loss: 0.09248060356511784, Accuracy: 0.9698405041464816
Epoch 7, Loss: 0.08873785741384871, Accuracy: 0.9701638065522621
Epoch 8, Loss: 0.08554383677378409, Accuracy: 0.9698533336070285
Epoch 9, Loss: 0.083543563476011, Accuracy: 0.9703382872156991
Epoch 10, Loss: 0.08140952343911902, Accuracy: 0.9705332950160112
```

Figure 5: Epochs of Transformer Model

former model has been run for 10 epoch runs where we can see that the accuracy has gone up to 97.05

6.4 Bert Model

The evaluate model function is defined to assess the model's performance on the test set. It outputs predictions, which are then used to calculate the accuracy and generate a detailed classification report. The Name Entity Recognition model by using Bert model has been run for 10 epoch runs where we can see that the accuracy has gone up to 97.94

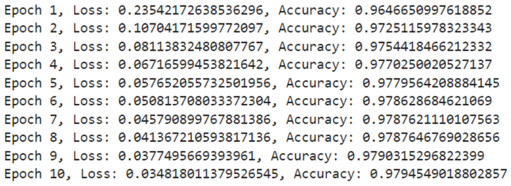
7 Conclusion

- The BiLSTM model achieved a respectable accuracy of 80.77 Percentage , showing its capability in sequence modeling for NER.

- The Transformer model significantly outperformed the BiLSTM with an accuracy of 97.05 Percentage, showcasing the power of self-attention mechanisms in capturing long-range dependencies.

- The CNN model presented competitive results with an accuracy of 96.92 Percentage, indicating its efficacy in feature extraction for NER.

- The BERT model topped the performance chart with a stellar accuracy of 97.95 Percentage, demonstrating the state-of-the-art capabilities of contextual embeddings in NER.



```
Epoch 1, Loss: 0.23542172638536296, Accuracy: 0.9646650997618852
Epoch 2, Loss: 0.10704171599772097, Accuracy: 0.9725115978323343
Epoch 3, Loss: 0.08113832480807767, Accuracy: 0.9754418466212332
Epoch 4, Loss: 0.06716599453821642, Accuracy: 0.9770250020527137
Epoch 5, Loss: 0.057652055732501956, Accuracy: 0.9779564208884145
Epoch 6, Loss: 0.050813708033372304, Accuracy: 0.978628684621069
Epoch 7, Loss: 0.045790899767881386, Accuracy: 0.9787621110107563
Epoch 8, Loss: 0.041367210593817136, Accuracy: 0.9787646769028656
Epoch 9, Loss: 0.0377495669393961, Accuracy: 0.9790315296822399
Epoch 10, Loss: 0.034818011379526545, Accuracy: 0.9794549018802857
```

Figure 6: Epochs of BERT Model.

- The study reveals a clear trend towards the superior performance of models like Transformers and BERT that leverage deep learning advances, suggesting a pivot away from traditional RNN-based architectures for NER tasks.

8 Contributions for each team member

8.1 Naveenreddy Narayana

Handling Data Preprocessing Tasks:
I am responsible for sourcing the dataset for Named Entity Recognition (NER) tasks. This involves accessing the dataset from its source.
Once the dataset is obtained, I performed data cleaning tasks to ensure the data is consistent, accurate, and free of errors or inconsistencies. This involved removing duplicates, handling missing values, correcting errors in labeling, and standardizing data formats.
Then transformed the raw data into a format suitable for training the machine learning model. This includes tokenization, vectorization, normalization, and other preprocessing techniques to prepare the data for input into the NER model.
Collaborating with Jaya prakash and Sai Narendra:
I worked closely with Jaya prakash to ensure that the preprocessed data is seamlessly integrated into the machine learning model. This involved coordinating the data preprocessing tasks with the model development process to ensure compatibility and consistency.
I maintained open communication with Jayaprakash and Sai Narendra to discuss any challenges, updates, or issues related to data preprocessing and model integration. This collaborative approach ensured that both team members are aligned and working towards the same goals with me.

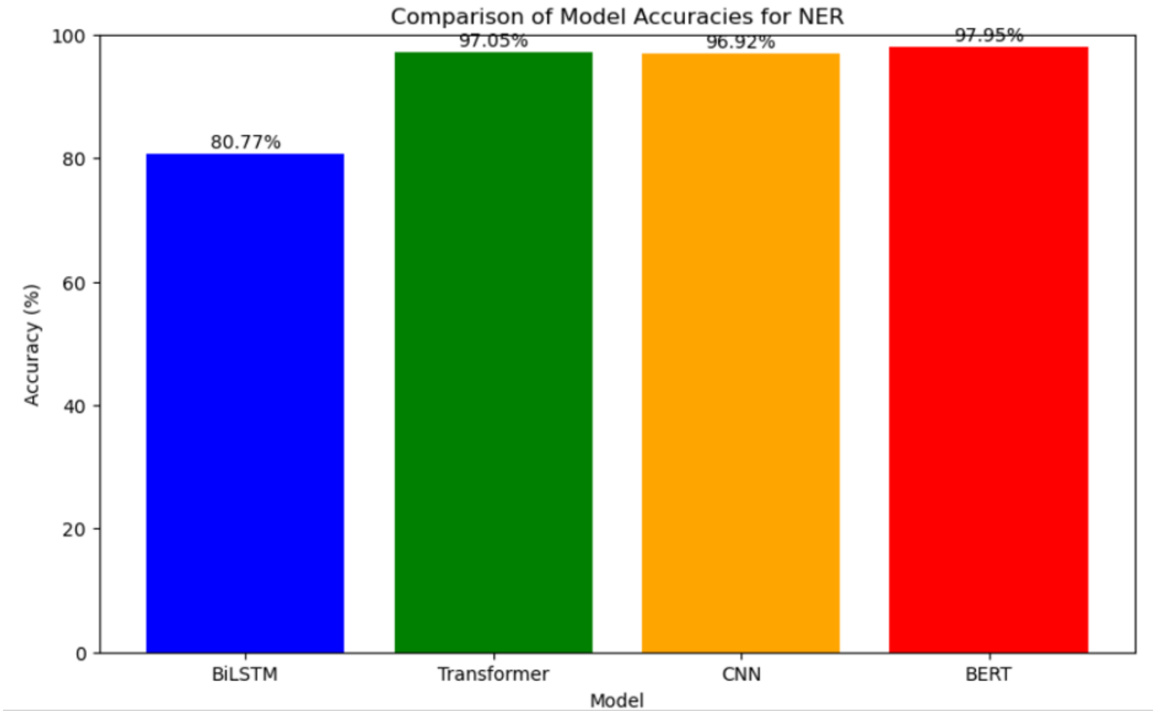


Figure 7: Comparision of the results of All Models.

8.2 Jayaprakash Narayana Raavi

Developing and Fine-Tuning the NER Model:
I am responsible for selecting the appropriate machine learning techniques and algorithms for developing the NER model. This involves assessing the requirements of the task, evaluating the strengths and weaknesses of different approaches, and choosing the most suitable model architecture.
I implemented the selected model architecture using machine learning libraries such as TensorFlow, PyTorch, or scikit-learn. This includes building the neural network layers, defining the loss function, configuring optimization algorithms, and fine-tuning hyperparameters to optimize model performance.
I continuously monitored and evaluated the performance of the NER model during training and testing phases.I also identified areas for improvement, troubleshooted issues such as overfitting or underfitting, and implemented strategies to enhance model accuracy, precision, and recall.
Collaborating with Naveenreddy and Sai Narendra:
I collaborated closely with Naveenreddy and Sai to ensure that the preprocessed data is effectively integrated into the NER model. This involves verifying data compatibility, performing data validation checks, and troubleshooting any issues that arise during the integration process.
I also provided feedback to Naveenreddy and sai on the quality and suitability of the preprocessed data for training the model. He also iterated on model development based on insights gained from analyzing the data and evaluating model performance.

8.3 Sai Narendra Reddy Lakkireddy

Assistance with Data Preprocessing:
I have assisted Naveenreddy with data preprocessing tasks, such as loading, cleaning, and transforming data. This involved collaborating closely with Naveenreddy to ensure that the dataset is prepared effectively for training the NER model.
Supported in Model Development and Fine-Tuning:
I contributed to the development and fine-tuning of the NER model alongside Jayaprakash. This could involve assisting with algorithm selection, model implementation, and optimization strategies to enhance model performance.
Documentation and Reporting:
I also worked on documenting the project's progress, including detailing the data preprocessing steps, model development process, and experimental results. I engaged in collaborative communication with both Naveenreddy and Jayaprakash, providing support and feedback as needed throughout the project lifecycle. This involved regular meet-

ings, discussions, and updates to ensure alignment and progress towards project goals.

9 References

Jana Strakova and Milan Straka and Jan Hajic. Neural Architectures for Nested NER through Linearization. Charles University. Faculty of Mathematics and Physics. Institute of Formal and Applied Linguistics

Bill Yuchen Lin and Wei Lu. 2018. Neural adaptation layers for cross-domain named entity recognition. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2012–2022. Association for Computational Linguistics.

Qi Liu, Yue Zhang, and Jiangming Liu. 2018. Learning domain representation for multi-domain sentiment classification. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Long Papers), volume 1, pages 541–550. Association for Computational Linguistics.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Long Papers), volume 1, pages 1064–1074. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In Advances in Neural Information Processing Systems, pages 3111–3119. Lili Mou, Zhao Meng, Rui Y.

Lample, Guillaume, et al. ”Neural Architectures for Named Entity Recognition.” Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2016.

Peters, Matthew E., et al. ”Deep contextualized word representations.” Proceedings of NAACL-HLT. 2018.

Devlin, Jacob, et al. ”BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” Proceedings of NAACL-HLT. 2019. Liu, Yang, et al. ”RoBERTa: A Robustly Optimized BERT Approach.” arXiv preprint arXiv:1907.11692. 2019.

Zhang, Yuhao, et al. ”ERNIE: Enhanced Representation through Knowledge Integration.” arXiv preprint arXiv:1904.09223. 2019.

Vaswani, Ashish, et al. ”Attention is All You Need.” Advances in Neural Information Processing Systems. 2017.

Wang, Alex, et al. ”BERTweet: A pre-trained language model for English Tweets.” arXiv preprint arXiv:2005.10200. 2020.

Akbik, Alan, et al. ”FLAIR: An easy-to-use framework for state-of-the-art NLP.” Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations. 2019.

Sang, Erik F. T. K. ”Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition.” Proceedings of CoNLL-2003. 2003. Chiu, Jason PC, and Eric Nichols. ”Named entity recognition with bidirectional LSTM-CNNs.” Transactions of the Association for Computational Linguistics. 2016.

Huang, Zhiheng, et al. ”Bidirectional LSTM-CRF models for sequence tagging.” arXiv preprint arXiv:1508.01991. 2015.

Bordes, Antoine, et al. ”Translating embeddings for modeling multi-relational data.” Advances in Neural Information Processing Systems. 2013. Chen, Danqi, et al. ”Neural symbolic machines: Learning semantic parsers on freebase with weak supervision.” Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics. 2017.