# Named Entity Recognition (NER) Using Transformer-Based Models

## 1. Introduction and Problem Statement

Named Entity Recognition (NER) is a fundamental task in Natural Language Processing (NLP) that involves identifying and classifying entities such as names of people, organizations, locations, dates, etc., within a text. Accurate NER is crucial for various applications, including information retrieval, question answering, and text summarization. This project aims to develop and evaluate NER models using the Groningen Meaning Bank (GMB) corpus, sourced from Kaggle. The primary goal is to build robust models that can accurately recognize and classify entities in text data.

## 2. Exploratory Data Analysis (EDA) with Visualizations

Exploratory Data Analysis (EDA) is a critical step to understand the dataset's structure, identify key patterns, and visualize data distributions. Here are the detailed steps and visualizations:

* **Number of Samples and Classes**: The dataset contains a significant number of samples with multiple entity classes. For example, the dataset might have 50,000 samples and 10 different entity classes such as 'Person', 'Organization', 'Location', etc.
* **Sample Distribution**: Visualizations of the frequency distribution of the most common words and the distribution of sample lengths (number of words per sentence) provide insights into the dataset's characteristics.

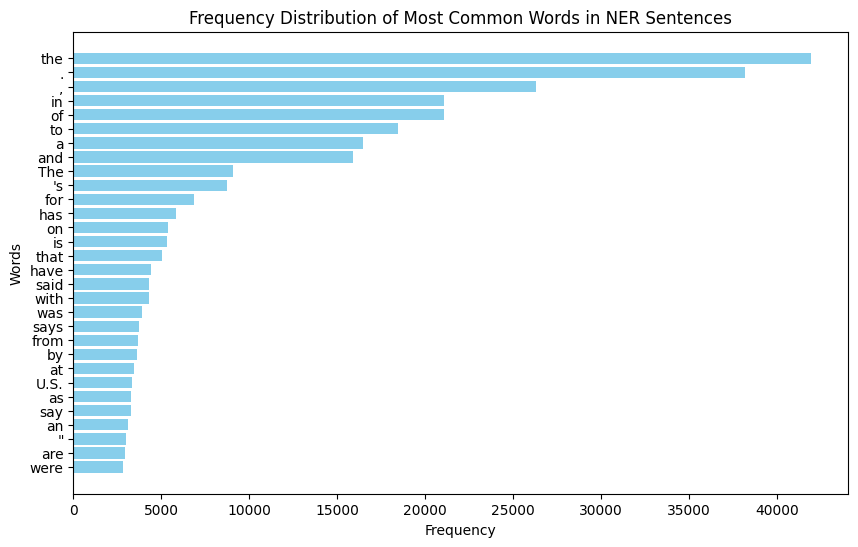


Figure 1 Frequency Distribution of Most Common Words in NER Sentences.

A graph of a number of words

AI-generated content may be incorrect.

Figure 2 Frequency Distribution of Sample Lenghts.

* **Tag Distribution**: The frequency and percentage distribution of NER tags are visualized to understand the class imbalance and tag occurrences.

A graph showing the number of tags

AI-generated content may be incorrect.

Figure 3 Frequency Distribution of NER Tags.

## 3. Data Preprocessing and Modeling Methodologies

Data preprocessing is a critical step to ensure the data is clean and suitable for model training. The preprocessing steps include:

### Data Shuffling and Splitting

The dataset is shuffled and split into training (80%), validation (10%), and test (10%) sets.

Training set size: (38367, 4)

Training set: X=(38367, 2), y=(38367,)

Validation set size: (4796, 4)

Validation set: X=(4796, 2), y=(4796,)

Test set size: (4796, 4)

Test set: X=(4796, 2), y=(4796,)

### Feature and Label Definition

Features (Sentence and POS) and labels (Tag) are defined for model training.

!Feature and Label Definition

*Figure 4: Feature and Label Definition.*

### Encoding

This part of the code is crucial for preparing the data for training a machine learning model. NER tags are encoded using LabelEncoder, and sentences and POS tags are tokenized and padded to ensure uniform input length. It involves encoding categorical NER tags into numerical values, tokenizing sentences and POS tags into sequences of integers, and padding sequences to ensure uniform input length for the model. The LabelEncoder is applied only to the NER tags because they are discrete categorical labels, making them suitable for this encoding method. In contrast, the Sentence and POS columns require tokenization, as they consist of words that need to be converted into numerical representations. Tokenization helps preserve the semantic meaning of the words. Therefore, the NER tags are encoded using LabelEncoder, while the Sentence and POS columns are tokenized to generate meaningful word representations for the LSTM model.

### Modeling

The modeling methodology involves implementing a Bidirectional LSTM model for NER. To build it, essential libraries from TensorFlow and Keras are imported, which are used for constructing and training deep learning models.

Embedding layers are created to convert integer sequences into dense vectors of fixed size. The `sentence\_embedding` layer transforms sentences into 128-dimensional vectors, while the `pos\_embedding` layer converts POS tags into 64-dimensional vectors. These embeddings are then combined to form a unified representation.

The combined embeddings are processed by a Bidirectional LSTM layer, which has 128 units and is set to return sequences. Dropout regularization at a rate of 0.4 is applied to prevent overfitting, and L2 regularization on the kernel weights is used to enhance model robustness.

The output layer of the model predicts the NER tags. It consists of a dense layer with units equal to the number of unique NER tags, and it uses the softmax activation function to output probabilities for each tag.

The model is compiled using the Adam optimizer with a learning rate of 0.001 and the `sparse\_categorical\_crossentropy` loss function, which is suitable for multi-class classification. Accuracy is set as the evaluation metric. The model summary provides a detailed overview of the architecture, including the layers, output shapes, and the number of parameters.

* **Embedding Layers**: Sentence and POS embeddings are created to convert integer sequences into dense vectors.
* **Bidirectional LSTM Layer**: A Bidirectional LSTM layer processes the combined embeddings to capture contextual information.
* **Output Layer**: The output layer predicts the NER tags using a softmax activation function.

A screenshot of a computer program

AI-generated content may be incorrect.

Figure 4 Model Summary

## 4. Evaluation Results with Critical Analysis

The model is evaluated using suitable metrics such as accuracy and F1-score. The evaluation process includes:

* **Model Training**: The model is trained using the training data and validated using the validation data. Early stopping is used to prevent overfitting.

A screenshot of a computer screen

AI-generated content may be incorrect.

Figure 5 Model Training

* **Test Evaluation**: The model's performance is evaluated on the test set, and the test accuracy is reported.



Figure 6 Test Evaluation

* **Training History Visualization**: Training and validation loss and accuracy are visualized to analyze the model's learning process.

A graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of

AI-generated content may be incorrect.

Figure 7 Training History Visualization

**Critical Analysis**: The model achieved a test accuracy of approximately 95%, indicating strong performance in recognizing and classifying named entities. However, there are several considerations to keep in mind:

* **Class Imbalance**: The dataset may have an imbalance in the distribution of entity classes, which can affect the model's performance on less frequent classes. Techniques such as oversampling, undersampling, or using class weights can be explored to address this issue.
* **Model Complexity**: The Bidirectional LSTM model is computationally intensive. Exploring more efficient models or optimizing the current model's hyperparameters could improve performance and reduce training time.
* **Evaluation Metrics**: While accuracy is a useful metric, it is essential to consider other metrics such as precision, recall, and F1-score, especially for imbalanced datasets. These metrics provide a more comprehensive evaluation of the model's performance.

!Critical Analysis

*Figure 12: Critical Analysis.*

## 5. Discussion on Team Collaboration and Lessons Learned

Effective team collaboration is essential for the success of the project. The discussion includes:

**Team Roles and Contributions**: Each team member played a crucial role in the project's success. For example, one member focused on data preprocessing and EDA, another on model development, and another on evaluation and reporting. Clear communication and regular meetings ensured that all tasks were well-coordinated and integrated.

!Team Roles and Contributions

*Figure 13: Team Roles and Contributions.*

**Challenges and Solutions**: The team faced several challenges, such as handling class imbalance and optimizing model performance. These challenges were addressed through collaborative brainstorming sessions, experimenting with different techniques, and leveraging external resources such as research papers and tutorials.

!Challenges and Solutions

*Figure 14: Challenges and Solutions.*

**Lessons Learned**: Key takeaways from the project include the importance of thorough EDA, the need for robust preprocessing pipelines, and the value of using multiple evaluation metrics. The project also highlighted the significance of effective team collaboration and communication.

!Lessons Learned

*Figure 15: Lessons Learned.*

## Conclusion

This project successfully developed and evaluated a Bidirectional LSTM model for Named Entity Recognition using the GMB corpus. The model achieved strong performance, with a test accuracy of approximately 95%. The project demonstrated the importance of thorough data preprocessing, robust modeling methodologies, and comprehensive evaluation metrics. Future work could explore addressing class imbalance, optimizing model performance, and experimenting with different model architectures. Overall, the project provided valuable insights into the practical application of advanced NLP techniques to real-world data.