# Neural Dependency Parsing Mini Project Report

Ilayda Soz Yilmaz

February 5, 2025

## Introduction

The objective of this mini project is to implement a bilinear dependency parser using PyTorch. Dependency parsing is a critical task in natural language processing and in this work, a simple bilinear parser is constructed and trained on a UD dataset to predict the syntactic heads of tokens.

## Methodology

### Data Preparation

The dataset is derived from the English UD EWT training set provided in the CoNLL-U format. The preprocessing steps included:

- Loading sentences and their corresponding dependency annotations using the `conllu` library.
- Extracting tokens and their head indices.
- Converting the head indices from 1-based to 0-based indexing, and mapping the root (index 0) to a special value (-1).
- Building a vocabulary by counting token frequencies and reserving indices for special tokens such as `<PAD>` and `<UNK>`.
- Converting sentences into sequences of word indices.
- Padding the sequences to handle variable lengths during batching.

### Model Implementation

The dependency parser was implemented with the following components:

1. **Embedding Layer:** Converts word indices into dense vectors of dimension 128.
2. **Bidirectional LSTM:** Processes the sequence of embeddings to capture context-aware representations, using a hidden dimension of 256.
3. **MLP Layers:** Two separate linear layers generate representations for potential head words and dependent words.

4. **Bilinear Scoring Layer:** Computes scores for every potential head-dependent pair using a bilinear transformation.

The entire model was implemented in Python using PyTorch.

### Training

The training of the model was conducted using the following settings:

- **Embedding Dimension:** 128
- **Hidden Dimension:** 256
- **Batch Size:** Initially 64 (increased to 512 later)
- **Number of Epochs:** 10
- **Learning Rate:** 0.001 (using the Adam optimizer)
- **Loss Function:** Cross-entropy loss

During training, the model's performance was monitored through the average loss per epoch and its accuracy on the test set.

## Results

Initial experiments with the model show promising improvements in accuracy over training epochs:

- After 3 epochs: Test Accuracy $\approx$ 45.76%
- After 5 epochs: Test Accuracy $\approx$ 60.76%
- After 10 epochs: Test Accuracy reached **81.62%**

These results demonstrate that the bilinear dependency parser significantly benefits from additional training, with the final model showing a strong ability to capture syntactic dependencies.

## Discussion

The experimental results indicate that the implemented bilinear parser is capable of learning complex syntactic structures. Key points include:

- A clear improvement in performance with an increased number of epochs.
- While 81.62% accuracy is promising, further improvements could be attained by exploring more complex architectures. Like additional linguistic features (e.g., POS tags), or utilizing attention mechanisms.

# Conclusion

This project successfully implemented a bilinear dependency parser using PyTorch. The model progressively improved its performance, ultimately achieving a test accuracy of 81.62% after 10 epochs. These results confirm that even a relatively simple bilinear scoring approach is effective for dependency parsing.

# References


- Kiperwasser, E., & Goldberg, Y. (2016). Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations. *Transactions of the Association for Computational Linguistics.*

- Dozat, T., & Manning, C. D. (2017). Deep Biaffine Attention for Neural Dependency Parsing. *ICLR.*

- PyTorch Documentation. `https://pytorch.org/docs/`