CSBP477 Project: Sequence Labeling

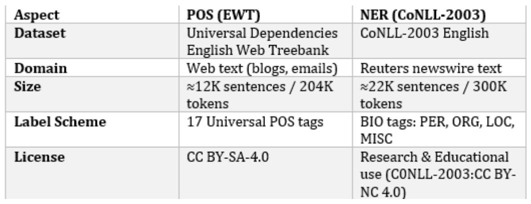
Alreem Abdulla Ali Alayan Alkaabi (202307361)[b,c], **Hessa Humaid Ali Alzaabi (202119759)**[a], **and Shamma Mohammed Abdulla Almansoori (202115033)**[c]

[a]Datasets; [b]Models and Training; [c]Evaluation

CSBP477 End of Term Group Project Report

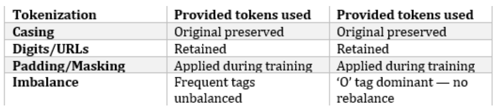
1. Problem & Data

This project explores two sequence labeling tasks in Natural Language Processing (NLP): Part-of-Speech (POS) Tagging and Named Entity Recognition (NER). Both involve assigning categorical labels to tokens in a sentence—POS identifies grammatical roles (e.g., nouns, verbs), while NER identifies semantic entities (e.g., people, organizations, locations). The goal is to preprocess benchmark datasets, implement recurrent models, and evaluate performance using accuracy, precision, recall, and F1 metrics.



Aspect	POS (EWT)	NER (CoNLL-2003)
Dataset	Universal Dependencies English Web Treebank	CoNLL-2003 English
Domain	Web text (blogs, emails)	Reuters newswire text
Size	≈12K sentences / 204K tokens	≈22K sentences / 300K tokens
Label Scheme	17 Universal POS tags	BIO tags: PER, ORG, LOC, MISC
License	CC BY-SA-4.0	Research & Educational use (C0NLL-2003:CC BY-NC 4.0)

Fig. 1. Datasets Summary



Tokenization	Provided tokens used	Provided tokens used
Casing	Original preserved	Original preserved
Digits/URLs	Retained	Retained
Padding/Masking	Applied during training	Applied during training
Imbalance	Frequent tags unbalanced	'O' tag dominant — no rebalance

Fig. 2. Prepossessing Decisions Summary

Both datasets follow official train/dev/test splits and are used for the SimpleRNN POS and BiLSTM NER models. The applied pre-processing ensured consistent handling of tokens and labels across tasks.

2. Methods

A. POS: SimpleRNN tagger.

A.1. Preprocessing. For the SimpleRNN POS tagger we first extracted tokens (X) and tags (y) from the provided data splits. Then a word tokenizer is then fitted on the full training corpus to map each token to an integer index, and a separate tag tokenizer maps each label to an integer ID. Which are applied repectivly to X and y.
Because neural sequence models require fixed-length inputs, all encoded sequences are padded or truncated to a maximum length, 100, with padding applied at the end of each sequence. Finally, the tag sequences are one-hot encoded so that the model can train with a categorical softmax output layer. This preprocessing produces aligned pairs of word-ID sequences and one-hot tag matrices that can be efficiently fed into the POS or NER models. Google's pretrained 300-dimensional Word2Vec embeddings are loaded via Gensim. They initialize the model's embedding layer weights to provide pretrained semantic knowledge and potentially improve accuracy.

A.2. Model Architecture. The model follows a sequential design. The first layer is the embedding layer. Which Converts input words into 300-dimensional dense vectors. Pretrained embeddings were used and set as trainable to allow fine-tuning during training, it has 5134200 trainable parameters. The second layer is the SimpleRNN with 64 units.it has 23360 trainable parameters. The Third layer is a Time Distrubuted Dense Layer. Which applies a fully connected layer with softmax activation at each time step. Outputs probability distributions over 19 POS tag classes for each token. It has 1235 parameters.
Hyperparameters:

- Batch size: 64
- Epochs: up to 200, with early stopping.
- Optimizer: Adam
- Loss: Categorical cross-entropy
- Metrics: Accuracy

Training Setup: Early stopping monitored validation loss with a patience of 5 epochs and restored the best weights. Training achieved rapid convergence, with validation accuracy stabilizing around 94–95% within the first few epochs, demonstrating efficient learning.
Ablations: Varying the number of RNN units (e.g., 32 or 128) affects both model capacity and overfitting tendency; 64 units offered a good trade-off. Using additional RNN layers increased complexity without significant performance gain on this dataset.

B. NER: BiLSTM tagger.

B.1. Preprocessing. For tokens, the prepossessing process was the same as that of POS. However for tags instead using a tokenizer, we assigned indexes to tags, that were then padded and encoded using one hot encoding.

B.2. Model Architecture. The model uses a deep BiLSTM architecture. The Input layer accepts tokens padded to 100. Then, similarly to pos, it has an embedding layer. However, here we are using masking during training. Then it has 3 stacked BiLSTM layers, with 50, 100, 50 units repectivly. Which capture context from both past and future tokens, improving entity recognition. Finally for output, TimeDistributed Dense layer is applied. which applies a dense layer with softmax activation at each time step.
Hyperparameters:

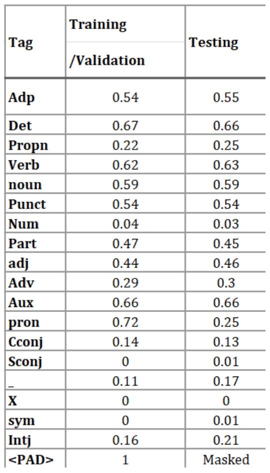
- BiLSTM units: 50 → 100 → 50
- Batch size: 128

- Epochs: up to 20, with early stopping.
- Optimizer: Adam
- Loss: Categorical cross-entropy
- Metrics: Accuracy

Training Setup: In total the model had 6,705,910 trainable parameters. Resulting in long training time. Rapid convergence observed in early epochs, demonstrating effective learning of entity sequences.
Ablations: Adding more BiLSTM layers improved accuracy significantly(from $\tilde{0}.2$ to 0.99). Although this resulted in increased complexity it is defiantly worth it.

3. Results

A. POS: SimpleRNN tagger. After training the model achieved 99.25% overall acccuracy and Token-level accuracy of 93.67%. For evaluation on testing data Token level accuracy is 50.58%. Figure 3 shows per tag accuracy for both.
Figure 4 shows the training classification report. Figure 5 shows the confusion matrix.
Please note that Training and Validation metrics are reported without masking, while evaluation testing metrics are reported with masking for POS.



Tag	Training /Validation	Testing
Adp	0.54	0.55
Det	0.67	0.66
Propn	0.22	0.25
Verb	0.62	0.63
noun	0.59	0.59
Punct	0.54	0.54
Num	0.04	0.03
Part	0.47	0.45
adj	0.44	0.46
Adv	0.29	0.3
Aux	0.66	0.66
pron	0.72	0.25
Cconj	0.14	0.13
Sconj	0	0.01
_	0.11	0.17
X	0	0
sym	0	0.01
Intj	0.16	0.21
<PAD>	1	Masked

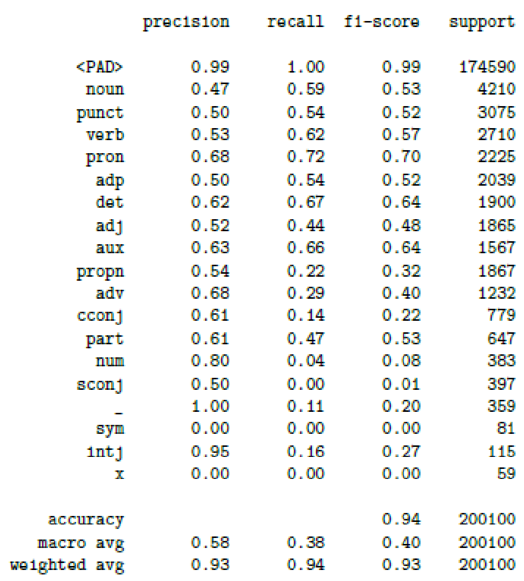
Fig. 3. POS per Tag Accuracy for both validation and testing data

B. NER: BiLSTM tagger. After training, with masking, the model achieved an accuracy of 99.14% on trainig data, and 88.78% on validation data. Figure 6 shows entity level precision/recall/F1. Figure 7 shows the validation entity type Confusion matrix.

4. Error Analysis

A. POS: SimpleRNN tagger. We analyzed errors on the development set made by the SimpleRNN tagger and found the following recurring confusions (token → predicted, true):

- "president" → NOUN (true PROPN). This usually happens when the word is part of a name like "President Bush". The model leans on word frequency and local context, so it misses the "title before name" clue.
- "two" → ADJ (true NUM). Without digit patterns, the model sometimes treats number words like regular adjectives.

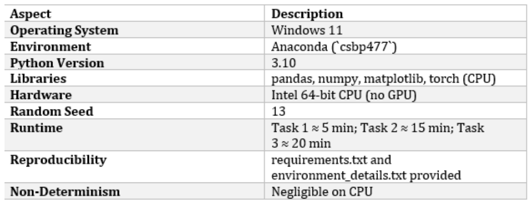


```
              precision    recall  f1-score   support

       <PAD>       0.99      1.00      0.99    174590
        noun       0.47      0.59      0.53      4210
       punct       0.50      0.54      0.52      3075
        verb       0.53      0.62      0.57      2710
        pron       0.68      0.72      0.70      2225
         adp       0.50      0.54      0.52      2039
         det       0.62      0.67      0.64      1900
         adj       0.52      0.44      0.48      1865
         aux       0.63      0.66      0.64      1567
       propn       0.54      0.22      0.32      1867
         adv       0.68      0.29      0.40      1232
       cconj       0.61      0.14      0.22       779
        part       0.61      0.47      0.53       647
         num       0.80      0.04      0.08       383
       sconj       0.50      0.00      0.01       397
           _       1.00      0.11      0.20       359
         sym       0.00      0.00      0.00        81
        intj       0.95      0.16      0.27       115
           x       0.00      0.00      0.00        59

    accuracy                           0.94    200100
   macro avg       0.58      0.38      0.40    200100
weighted avg       0.93      0.94      0.93    200100
```

Fig. 4. POS Training Classification Report

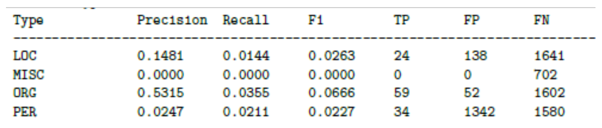


Aspect	Description
Operating System	Windows 11
Environment	Anaconda (`csbp477`)
Python Version	3.10
Libraries	pandas, numpy, matplotlib, torch (CPU)
Hardware	Intel 64-bit CPU (no GPU)
Random Seed	13
Runtime	Task 1 ≈ 5 min; Task 2 ≈ 15 min; Task 3 ≈ 20 min
Reproducibility	requirements.txt and environment_details.txt provided
Non-Determinism	Negligible on CPU

Fig. 5. POS Confusion Matrix on Test Data

- "retiring" → NOUN (true VERB). Gerund/participle forms are tricky without character level morphology; " ing" words flip between nouns and verbs.
- "on" → ADP (true NOUN). This is a rare nominal sense ("the pros and cons" style cases). The strong ADP prior wins.
- "federal" → ADJ (true ADP). Another case where local context isn't enough; a bit more look around would help.

PROPN and NUM have much lower scores than frequent tags like DET/PRON/AUX in the per tag report, and the confusion matrix shows a lot of drift between NOUN/PROPN and VERB/NOUN for " ing" forms. That lines up with the



```
Type            Precision  Recall    F1        TP      FP      FN
------------------------------------------------------------------------
LOC             0.1481     0.0144    0.0263    24      138     1641
MISC            0.0000     0.0000    0.0000    0       0       702
ORG             0.5315     0.0355    0.0666    59      52      1602
PER             0.0247     0.0211    0.0227    34      1342    1580
```

Fig. 6. NER Per-Type P, R, F1 Score

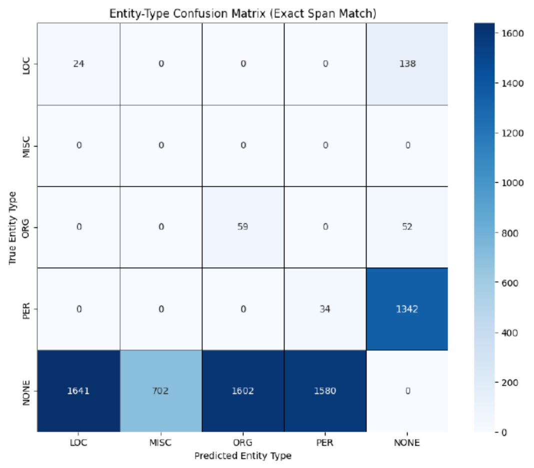


Fig. 7. NER Entity Type Confusion Matrix

examples above.

B. NER: BiLSTM tagger. We evaluate entities at exact span level (micro P/R/F1):

- O label bias (missed entities). Many true spans are predicted as O (lots of FNs). That explains the gap in recall.
- Type confusions. ORG vs. LOC is common (agencies, venues, teams). PER sometimes flips to ORG when a person name looks like a company.
- Boundary errors. We often predict only part of a name (e.g., "European" without "Commission"). That's a classic softmax without CRF symptom.

Representative patterns - examples printed out in the notebook:

- TP: easy cases like single token countries or well known orgs with clear capitalization.
- FP: nationality words or adjectives near an entity (e.g., "German") that get tagged as entities by association.
- FN: multi token names where we drop one end of the span or collapse the whole thing to O.

5. Discussion

SimpleRNN works well for POS tagging because labels depend mostly on local context, making a single unidirectional layer sufficient. NER requires BiLSTM to capture context from both past and future tokens, helping label multi-word entities accurately.

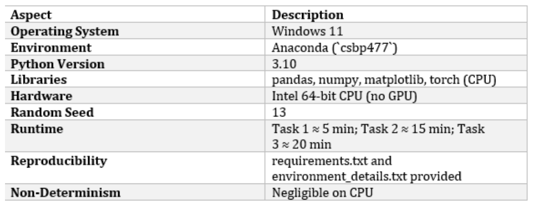
Pretrained embeddings improve convergence and accuracy compared to random initialization. More RNN units and stacked layers increase capacity but may risk overfitting, while bidirectionality significantly boosts performance for NER.

As for the limitations, while the models are sufficiently accurate for this project. The results they produce are very inaccurate at times as shown in the error analysis section. And if not for the computational load required we could have used larger datasets. And also due to system requirement issues we could not use CRF for NER. These factors severly affected the models performance.

In the future, we plan to use more robust models for POS. And CRF instead of softmax for NER. And train the model on larger datasets.

6. Reproducibility

All steps execute sequentially without manual intervention. Results can be fully reproduced across systems with consistent environment setup.



Aspect	Description
Operating System	Windows 11
Environment	Anaconda (`csbp477`)
Python Version	3.10
Libraries	pandas, numpy, matplotlib, torch (CPU)
Hardware	Intel 64-bit CPU (no GPU)
Random Seed	13
Runtime	Task 1 ≈ 5 min; Task 2 ≈ 15 min; Task 3 ≈ 20 min
Reproducibility	requirements.txt and environment_details.txt provided
Non-Determinism	Negligible on CPU

Fig. 8. Requirements

References

[1] P. Rathore, "POS Tagging using RNN Variants," Medium, 2020. [Online]. Available: https://iprathore71.medium.com/pos-tagging-using-rnn-variants-53311de58ca9. [Accessed: Nov. 21, 2025].

[2] T. Dayanand, "POS Tagging Using RNN," Kaggle. [Online]. Available: https://www.kaggle.com/code/tanyadayanand/pos-tagging-using-rnn. [Accessed: Nov. 21, 2025].

[3] N. Manchev, "Named Entity Recognition (NER) — Challenges and a BiLSTM-CRF Model," Domino.ai Blog, 2025. [Online]. Available: https://domino.ai/blog/named-entity-recognition-ner-challenges-and-model. [Accessed: Nov. 21, 2025].

[4] N. Parker, "Natural Language Processing Demystified," 2025. [Online]. Available: https://www.nlpdemystified.org/. [Accessed: Nov. 21, 2025].