Assignment 2 – Window-based Tagging

Daniel Bazar 314708181

Peleg shefi 316523638

Part 3 – Adding the external word embeddings to the tagger

Architecture

- Both tasks implement the same Network: an MLP with:
  - one hidden layer
  - tanh activation function.
- The network trained with a cross-entropy loss.
- We Experimented using a grid search function (called “param_search”) with several network configurations and chose the best configuration based on the DEV accuracy.
- The optimizer is Adam.
- We initialized the layers using Xavier.

Best parameters

- NER:
  - Hidden layer size: 130
  - Dropout probability: 0.4
  - Batch size: 64
  - Optimizer: Adam (Learning rate: 1e-3)
  - Epochs: 5
- POS:
  - Hidden layer size: 130
  - Dropout probability: 0.2
  - Batch size: 2048
  - Optimizer: Adam (Learning rate: 1e-3)
  - Epochs: 8

Considerations

- We handle words that are not in the “vocab.txt” file by assigning them the UNK token. Just like we did in part 1.
- When using the pre-trained embedding vectors, we transform the training and the dev data to **lower case** because the embedding vocabulary is lower-case.
- The embedding vocabulary **contains special words** like “DGDGDGDG”, “DG.DG”, “+DG”, “NNNUMMM”, etc. we treated those words as digits patterns.
- We padded the sentences with SOS (start of string) and EOS (end of string) at the beginning and end of the sentence. We tried doing SOS2, SOS1 and EOS1, EOS2 and by that capturing more that but it didn’t improve our results.
- It seems that in this part as well **the Xavier initialized** is a key to getting good results. It even outperformed the pre-trained embedding.

Results

Accuracy comparison: The accuracy didn't improve compared to the results in part 1, which didn't use pre-trained embeddings.

- NER:
    - Loss validation: 0. 194
    - Accuracy validation: 79.34%

  The accuracy dropped!

- POS:
    - Loss validation: 0.161
    - Accuracy: 95.1%

  No significant changes.

Graphs