# Assignment 2 – Window-based Tagging

Daniel Bazar 314708181

Peleg shefi 316523638

## Part 4 – Adding sub-word units

### Architecture

* Both tasks implement the same Network: MLP
  + Embedding layer - **sum of word embedding and sub-word embedding**.
  + one hidden layer
  + tanh activation function.
* The way we implemented the sub-word embedding:
  + Assign an embedding vector to each prefix and suffix of 3 letters. (Among the prefixes and suffixes that were observed in the corpus).
  + Then, we represent each word as **the sum of the word vector, the prefix vector, and the suffix vector.**
* The network trained with cross-entropy loss.
* The optimizer is Adam.
* We Experimented with several network configurations and chose the best configuration based on the DEV accuracy using our grid search function.

### Best parameters

* **NER:**
  + Hidden layer size: 130
  + Dropout probability: 0.4
  + Batch size: 128
  + Optimizer: Adam (Learning rate: 0.001)
  + Epochs: 5
* **POS:**
  + Hidden layer size: 130
  + Dropout probability: 0.4
  + Batch size: 2048
  + Optimizer: Adam (Learning rate: 0.001)
  + Epochs: 6

### Considerations

* The sub-word unit method can be combined with the pretrained embedding so we had to consider all the things we considered in the pretrained part:
  + We handle unseen the same way as in the other parts (with the UNK token).
  + If we used the pre-trained embedding vectors, we transform the training and the dev data to lower case because the embedding vocabulary being lower-case.
  + Because the embedding vocabulary contains special words like “DGDGDGDG”, “DG.DG”, “+DG”, “NNNUMMM”, etc. We treated those words as digit patterns.
  + We padded the sentences with SOS (start of string) and EOS (end of string) at the beginning and end of the sentence.
* If a length of a word is less than 3, we take the whole word as a suffix and prefix.

### Results

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **POS** | | **NER** | |
|  | With pre-trained | Without pre-trained | With pre-trained | Without pre-trained |
| **BATCH** | 128 | 2048 | 128 | 128 |
| **EPOCHS** | 6 | 3 | 5 | 3 |
| **HIDDEN LAYER** | 130 | 130 | 130 | 130 |
| **LR** | 0.001 | 0.001 | 0.001 | 0.001 |
| **DROPOUT** | 0.2 | 0.2 | 0.4 | 0.4 |
| **LOSS** | 0.145 | 0.136 | 0.191 | 0.158 |
| **ACCURACY** | 95.6 | 95.84 | 80.12 | 78.97 |

\*the results are the best we achieved and aren’t the same as we used in the comparison below.

\*\* All used subword

### Brief analysis of the results

In order to compare the performance of our model with the different embedding settings combinations [standard/pre-trained & sub-word/sub-word/pre-trained] we had to run them with the same parameters. We used the standard – without both as the baseline.

we run the model for 8 epochs with the following configuration:   
{'hidden\_layer': 130, 'dropout\_p': 0.4, 'batch\_size': 128, 'lr': 0.0004}

|  |  |  |
| --- | --- | --- |
| **Tagger** | **Best DEV Accuracy** | **conclusion** |
| standard | 82.95 | Baseline |
| Pre-trained & Sub-word | 79.54 | significant difference compared to the **baseline** |
| Sub-word | 79.25 | No significant difference compared to **pre trained** and **sub-word** |
| Pre-trained | 77.51 | significant difference compared **sub-word** |

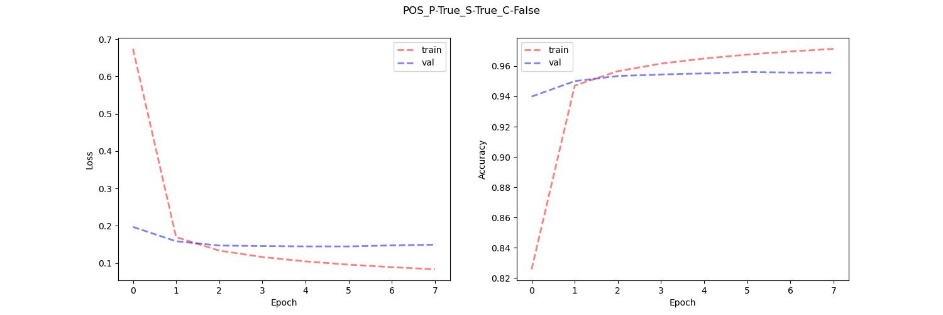
Main points:

* We can see that all of the methods are less optimal than our standard tagger which uses a Xavier initialization for the embedding.
* Although, it seems like the sub-word units are more useful than the pre-trained embedding.
* There is no significant difference in combining them together.
* Prefixes and suffixes can be helpful in two major ways:
  + For POS tagging- can tell the POS for example - ing,ify,ise are usually verbs. For ner not too much.
  + For unknown words that have a common prefix/suffix we get a better representation.

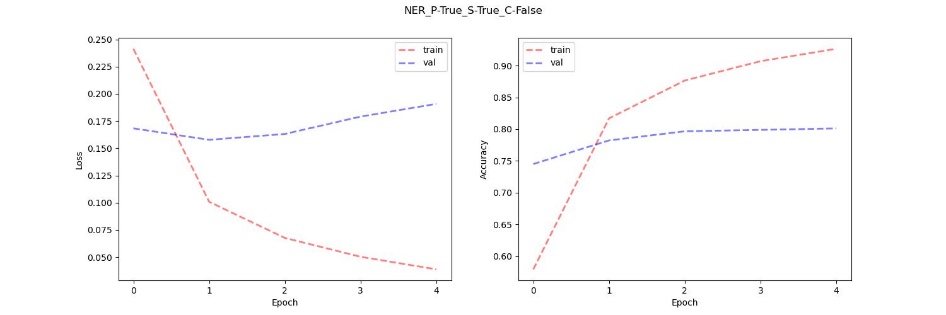
\*We didn’t have time to try it but using the prefix/suffix only on unknown words and having all the other words with the Xavier init may result in better performance.

### Graphs

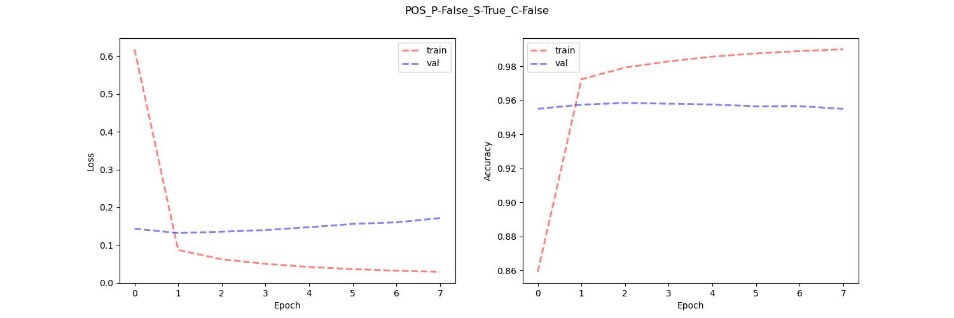
With pretrained embedding:



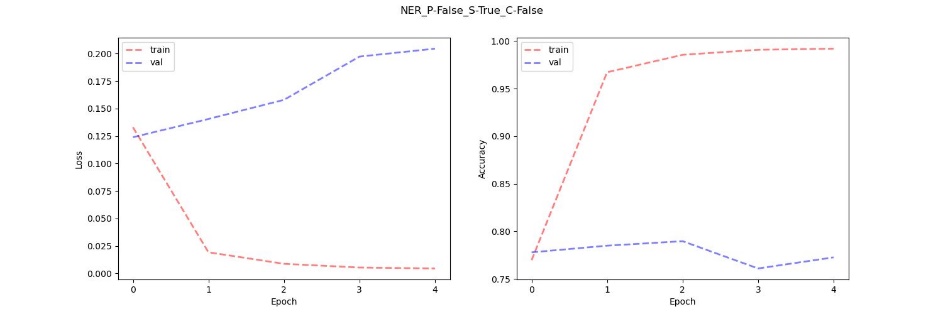
**POS with pre-trained & sub-word**



**NER with pre-trained & sub-word**



**POS with sub-word only**



**NER with sub-word only**