# Assignment 2 – Window-based Tagging

Daniel Bazar 314708181

Peleg shefi 316523638

## Part 4 – Adding sub-word units

### Architecture

* Both tasks implementing the same Network: MLP with one hidden layer and a tanh activation function. The MLP feeded from a sub-word embedding
* The way we implemented the sub-word embedding is to assign an embedding vector to each prefix of 3 letters and each 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 a cross-entropy loss
* We Experimented with several network configurations and chose the best configuration based on the DEV accuracy

### Best parameters

* NER:
  + Hidden layer size: 130
  + Dropout probability: 0.3
  + Batch size: 128
  + Optimizer: Adam (Learning rate: 1e-4)
  + Epochs: 6
* POS:
  + Hidden layer size: 90
  + Dropout probability: 0.2
  + Batch size: 64
  + Optimizer: Adam (Learning rate: 5e-05)
  + Epochs: 8

### Considerations

* The sub-word units method can be combined with the pretrained embedding so we had to consider all the things we take into account in the pretrained part:
  + We handle words that appear in the training file but not in the embedding file as we handle those words in part1 (words that don’t appear in dev file), we assign them 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 digits 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 of 3 we take the whole word as a suffix and prefix

### Results

* NER:
  + Loss validation: 0.101
  + Accuracy validation: 83.13%
* POS:
  + Loss validation: 0.135
  + Accuracy: 95.84%

### Brief analysis of the results

To check the performance of the methods we experiment the same configuration with pretrained embedding and sub-word units, with pretrained embedding and no sub-word units, with no pretrained embedding and sub-word units. As a baseline we train a model without any of them (no pretrained and no sub-word units).  
we test on the NER task (because it’s more difficult, The POS is always ~95%).  
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 | worser |
| Sub-word | 79.25 | No significant difference |
| Pre-trained | 77.51 | worst |

To sum up:

* we can see that all methods are harming our standard tagger.
* 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

### Graphs

With pretrained embedding:

With no pretrained embedding: